A network based motion control system for fountain application

Deng, Wu

2007

Deng, W. (2007). Network based motion control system for fountain application. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/39157>

<https://doi.org/10.32657/10356/39157>

9530273

A Network Based Motion Control System for Fountain Application

Deng Wu

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Master of engineering

2007

Acknowledgement

I would like to express my sincere gratitude and thanks to my supervisor and mentor Associate Professor, Dr. Low Kay Soon for his guidance, support and encouragement throughout this project. He has always encouraged me to try new ideas and concepts, and guided me on the research and implementation throughout my master's study.

I would like to thank Mr. Cao Runzi and Mr. Zhuang Hualiang for their valuable discussions and encouragement. I have learnt a lot from them through the discussions with them. I would also like to thank the Control Engineering laboratory technicians, Ms. Joanne Lim and Ms. Janet Tan, for their patience and assistance throughout the project.

Last but not least, I would like to thank OASE Asia Pacific Pte Ltd for funding this research.

Table of Contents

ACKNOWLEDGEMENTI

TABLE OF CONTENTS II

SUMMARY..... VI

LIST OF FIGURES VII

LIST OF TABLESXI

CHAPTER 1 INTRODUCTION 1

1.1 Motivation and Overview..... 1

1.2 Objectives..... 4

1.3 Main Contributions 5

1.4 Thesis Organization 6

CHAPTER 2 A REVIEW OF MULTI-AXIS MOTION CONTROL 8

2.1 Introduction 8

2.2 Performance Measures of Multi-axis motion system 9

2.3 Control Approaches of Multi-axis System 11

2.4 Conventional Multi-Axis Control 12

2.5 Contour Based Cross-coupled Control..... 13

2.6 Trajectory Coordinate Based Cross-Coupled Control..... 16

2.7 Integrated Cross-Coupled Control Approach..... 19

2.8 Conclusions 20

CHAPTER 3 ROTARY BIAXIAL CROSS-COUPLED CONTROL 21

3.1 Rotary Biaxial Motion System 21

3.2 Rotary Biaxial Cross-Coupled Control 23

3.3 Comparison between the proposed and direct approximation methods 26

3.4 Trajectory Generation 28

3.5 Experimental Setup and System Identification 33

3.6	Simulation Results	36
3.7	Experimental Results	38
3.7.1	Performance of Different Control Methods	39
3.7.2	Performance at Different Speeds	40
3.7.3	Performance at Different Trajectory Radius	41
3.7.4	Performances at Different Normal to Tangential Gain Ratios	43
3.8	Conclusions	45
CHAPTER 4 NETWORK SYSTEM DESIGN		46
4.1	Introduction	46
4.2	Industrial Network and Controller Area Network.....	47
4.2.1	Industrial Networks.....	47
4.2.2	Controller Area Network	49
4.3	Network System Structure	54
4.4	Message Encoding, Decoding and Scheduling	57
4.4.1	Message Encoding and Decoding	58
4.4.2	Priority Based Message Scheduling.....	58
4.5	Motion Data Streaming	61
4.5.1	Trajectory Down-Sampling.....	62
4.5.2	Motion Data Scheduler	63
4.5.3	Motion Data Sequence Control.....	64
4.5.4	Motion Data Buffering.....	66
4.5.5	Trajectory Up-Sampling	69
4.6	Motion Synchronization	70
4.7	Conclusions	72
CHAPTER 5 CONTINUOUS CLOCK STATE AND CLOCK RATE SYNCHRONIZATION		
73		
5.1	Introduction	73
5.2	The Clock Model.....	76
5.3	Evaluation of Existing Clock Adjustment Methods	79

5.3.1	Instantaneous Clock Adjustment.....	79
5.3.2	Amortization Based Clock Adjustment.....	80
5.3.3	Limitations of the Methods	81
5.4	The Proposed Continuous Integrated Clock Synchronization.....	82
5.4.1	Continuous Clock State Synchronization by Clock Rate Adjustment	82
5.4.2	Clock Rate Synchronization.....	83
5.4.3	Integrated Continuous Clock State and Clock Rate Synchronization	83
5.5	Precision Analysis	85
5.5.1	Clock State Synchronization with Instantaneous Clock Adjustment	85
5.5.2	Integrated Clock State and Clock Rate Synchronization	87
5.6	Implementation and Experimental results	89
5.7	Conclusions	94
CHAPTER 6 PROTOTYPING AND IMPLEMENTATION		95
6.1	Overview	95
6.2	Design of the Biaxial Motion Mechanism	97
6.3	Local Motion Controller.....	98
6.3.1	Hardware Design	98
6.3.2	Firmware Design and Implementation.....	101
6.4	Central Controller Design	107
6.4.1	Central Controller Software Overview	107
6.4.2	Graphical User Interface	108
6.4.3	Controller Block.....	110
6.4.4	Communication Interface.....	111
6.4.5	Trajectory Generation	111
6.5	Network System with Large Numbers of Nodes.....	112
6.6	Experimental Results with Water Spray	116
6.7	Conclusions	119

CHAPTER 7 CONCLUSIONS AND RECOMMENDATIONS..... 120

 7.1 Conclusions..... 120

 7.2 Recommendations for Future Work..... 121

REFERENCE 123

APPENDICES- 1 -

Summary

Modern fountains are becoming more sophisticated as people are pursuing better entertaining effect. More sophisticated technology, especially in the field of motion control and network systems, is demanded by the fountain industry to further expand the capability and feature of a fountain. In particular, a reliable network system is desired to support the synchronous motion control of a large number of multi-axis motion systems.

In this industrial funded research, multiple rotary type biaxial motion systems are used to drive the fountain nozzles. To improve the performance, a cross-coupled control method that transforms the spherical motion into local planar motion is proposed. Experiments have been conducted and verified that the proposed algorithm improves the contouring precision.

A network system for synchronous motion control has been designed based on the Controller Area Network (CAN). Application layer protocol has been developed to provide reliable data transmission. A novel continuous clock synchronization method by clock rate adjustment has been proposed to ensure synchronized motion among different biaxial motion systems. Analysis and experimental results indicate that the proposed clock synchronization method generates continuous clock and has a higher precision as compared to the conventional methods.

A prototype has been constructed with the proposed motion control algorithm and network protocol. The successfully implemented prototype has validated the overall performance of the system and shows much greater flexibility with higher system reliability.

List of Figures

Fig. 1-1 Illustration of the biaxial motion nozzle2

Fig. 1-2 Overview of the network based motion control system4

Fig. 2-1 (a) Performance measures (b) The trajectory coordinate 10

Fig. 2-2 Conventional biaxial motion control..... 12

Fig. 2-3 A typical cascaded motion control 13

Fig. 2-4 Contouring error of a linear biaxial motion system 14

Fig. 2-5 Contour based cross-coupled control for biaxial motion system 16

Fig. 2-6 Trajectory coordinate based cross-coupled control..... 18

Fig. 3-1 Error curves on the spherical surface22

Fig. 3-2 The approximated planar Cartesian coordinate on the sphere24

Fig. 3-3 Block diagram of rotary biaxial cross-coupled control26

Fig. 3-4 Illustration of different circular trajectories with the same radius29

Fig. 3-5 3D View of the circular contour (before transformation)31

Fig. 3-6 The trajectories of the circular contour (before transformation).....31

Fig. 3-7 3D View of the circular contour (after transformation)31

Fig. 3-8 The trajectories of the circular contour (after transformation).....31

Fig. 3-9 3D view of the dumbbell shape contour (before transformation)32

Fig. 3-10 The trajectories of the dumbbell shape contour (before transformation)...32

Fig. 3-11 3D view of the dumbbell shape contour (after transformation)33

Fig. 3-12 The trajectories of the dumbbell contour (after transformation).....33

Fig. 3-13 Block diagram of the experimental setup..... 34

Fig. 3-14 Photograph of the experimental setup.....34

Fig. 3-15 Time plot of input and output for θ axis system identification.....35

Fig. 3-16 Time plot of input and output for ϕ axis system identification35

Fig. 3-17 Step response of the two axis36

Fig. 3-18 Normal errors (simulation).....37

Fig. 3-19 Tangential errors (simulation).....37

Fig. 3-20 Normal error of circular contour39

Fig. 3-21 Tangential error of circular contour39

Fig. 3-22 Normal errors at 0.08m/s.....40

Fig. 3-23 Normal errors at 0.16m/s.....40

Fig. 3-24 Tangential errors at 0.08m/s.....40

Fig. 3-25 Tangential errors at 0.16m/s.....40

Fig. 3-26 ϕ axis trajectory of the profiles42

Fig. 3-27 $\cos \phi$ trajectory of the profiles.....42

Fig. 3-28 Normal errors with profile 1.....42

Fig. 3-29 Normal errors with profile 2.....42

Fig. 3-30 Tangential errors with profile 1.....42

Fig. 3-31 Tangential errors with profile 2.....42

Fig. 3-32 Normal errors with different gains44

Fig. 3-33 Tangential errors with different gains44

Fig. 4-1 Overview of the proposed network protocol.....47

Fig. 4-2 CAN message frame format.....51

Fig. 4-3 Hierarchical approach of network based control.....55

Fig. 4-4 Distributed approach of network based control56

Fig. 4-5 The proposed network system structure.....57

Fig. 4-6 Application data format.....58

Fig. 4-7 CAN identifier assignment.....60

Fig. 4-8 Motion data streaming mechanism62

Fig. 4-9 State diagram of sequence control mechanism65

Fig. 4-10 Buffer stack and indication levels68

Fig. 4-11 Experimental result of the managed buffer69

Fig. 5-1 Two stages of clock synchronization75

Fig. 5-2 The generation of physical clock76

Fig. 5-3 Illustration of discrete clock synchronization and amortization81

Fig. 5-4 Clock errors of the proposed method and conventional method.....92

Fig. 5-5 Clock errors at different synchronization intervals (proposed method).....93

Fig. 5-6 Clock errors at different synchronization intervals (conventional method).93

Fig. 6-1 Block diagram of the prototype.....96

Fig. 6-2 A picture of the prototype96

Fig. 6-3 The biaxial motion mechanism97

Fig. 6-4 Functional blocks of the controller board99

Fig. 6-5 The controller board prototype.....99

Fig. 6-6 Block diagram of the amplifier interface100

Fig. 6-7 Block diagram of the encoder interface101

Fig. 6-8 Flow chart of the main routine102

Fig. 6-9 Flow char of the motion control ISR.....104

Fig. 6-10 Flow chart of the PID calculation function105

Fig. 6-11 Flow chart of the CCC calculation function.....106

Fig. 6-12 Functional blocks of the central controller.....108

Fig. 6-13 The Graphical User Interface (GUI)109

Fig. 6-14 The trajectory generation field112

Fig. 6-15 Experimental setup with temporary waterproofing measures.....117

Fig. 6-16 Snaps shots of the experiments with water spay 118

List of Tables

Table 3-1 List of simulation parameters37

Table 3-2 Comparison of simulation errors with different control methods38

Table 3-3 Comparison of errors with different control methods39

Table 3-4 Comparison of errors at different speeds.....41

Table 3-5 Comparison of errors with different profiles.....43

Table 3-6 Comparison of errors with different tangential to normal gain ratio.....44

Table 4-1 List of priority levels of message types60

Table 4-2 List of feedbacks and Responses64

Table 5-1 Steady state clock errors at different synchronization interval.....93

Table 6-1 comparison of throughputs for 2 and 100 fountain heads system 113

Table 6-2 Comparison of throughputs with different nodes..... 115

Table 6-3 Comparison of requests for different nodes 116

Chapter 1

Introduction

1.1 Motivation and Overview

Modern fountains are the combination of art and technology that delivers the beauty of water through flows, motion, lights, and sounds. Technologies in motion control, networks, fluid dynamics and architecture are integrated to support the creative and diversified fountain designs. The increasing needs for greater flexibility and variety in fountain design has demanded more technology support. One of the latest demands is to incorporate multi-axis motion into the fountain nozzles, in order to obtain more freedom in the fountain pattern. A reliable network structure for fountain control and management is also desired to support the increased complexity of a modern large scale fountain.

This project aims to design a reliable network based motion control system that provides high flexibility and reliability for the use in a large scale fountain. The project has two important topics: the rotary biaxial motion system design and the network system design. The rotary biaxial motion system design includes the design of a rotary biaxial motion mechanism and the corresponding motion control algorithms for driving the fountain nozzles. The network design includes the development of network structure and protocol for controlling synchronously a large number of these biaxial motion systems. Brief introduction on these two topics are presented as follow.

In the present fountain technology, fountain nozzles have limited degree of freedom. They are either fixed in one position, or driven by a motor to swing or rotate in one direction. To increase its flexibility and degree of freedom, biaxial motion fountain nozzles are proposed. As shown in Fig. 1-1, the fountain nozzle is to be placed on a rotary biaxial motion system that is driven by two motors. In this way, the nozzle could have free movements on a half-spherical surface unlike the traditional fountain nozzle. Consequently, greater flexibility and variety in fountain design can be achieved. However, this requires more efforts on the controller design to ensure precise and reliable nozzle motion. Moreover, the movement of the nozzle should not cause entanglement of the water piping and wiring.

Fig. 1-1 Illustration of the biaxial motion nozzle

In this thesis, we investigate approaches for precision control of the rotary biaxial motion system. Although multi-axis motion system including biaxial drive is well known, most studies focus on linear multi-axis motions, whose individual axis is linear in motion. In this research, the biaxial drive for the fountain nozzle is rotary. Furthermore, the nozzle's motion is on a spherical surface instead of the planar surface. Consequently, it has different dynamics as compared to the conventional system. In this thesis, we propose a cross-coupled control algorithm for the system.

The proposed algorithm is also suitable for other applications that employ a rotary biaxial drive.

In a large-scale water fountain, there may be several tens to hundreds of fountain nozzles to be controlled simultaneously. A network system is used to connect them to a central controller for controlling, scheduling and coordinating their motions. Traditional nozzle has limited freedom and the control involves only simple on/off of the motor and pump. Hence, simple networks protocol such as RS485 and DMX512 can be sufficiently used. By incorporating biaxial-motion into the nozzle and demanding more complex synchronizing performance, higher network requirement is imposed. Consequently, a more advanced network system is to be considered to achieve the expectation.

In this thesis, we investigate a Controller Area Network (CAN) based system to fulfill the application requirements. The CAN standard is selected as the physical and data-link layer of the network. The application layer network protocol is also proposed. It is responsible for delivering the control command and motion data, scheduling the network traffic and synchronizing the motion of the nozzles. A novel integrated continuous clock state and clock rate synchronization algorithm is proposed and implemented in the network protocol. This clock synchronization algorithm ensures synchronous motion among different nozzles.

The integration of the rotary biaxial motion systems and the network forms the complete network based motion control system. As shown in Fig. 1-2, the system consists of the rotary biaxial motion mechanisms, local controllers, CAN based network and central controllers. All the central and local nodes are connected through the CAN bus. The proposed motion control algorithm is implemented in the local

controller. While the network protocols are implemented in both central and local controllers. By incorporating this system into fountain application, the flexibility, reliability and management of fountain will be greatly enhanced. A prototype system has been designed and constructed for evaluation.

Fig. 1-2 Overview of the network based motion control system

1.2 Objectives

1. To design a rotary biaxial motion system and propose a new cross-coupled control method to achieve high contouring motion accuracy. This is motivated by the intrinsic difference between rotary and linear based biaxial motion systems, and the fact that the existing cross-coupled control methods are based on the linear motion system.
2. To design a reliable network system for the application. A suitable industrial network is used as the physical and data-link layer for the network system. Network protocol for the application layer is to be proposed to properly deliver the real time command and motion data, as well as synchronize the motion of multiple nozzles.

3. To proposed a continuous clock synchronization method. This is motivated by motion synchronization, which requires the clock of each node to be synchronized and maintained continuous. Discontinuous clock is undesirable due to degradation in performance.
4. To design and build a prototype of the network based motion control system. With this prototype, experiments can be performed to evaluate the system and verify the proposed motion control algorithm and network protocol.

1.3 Main Contributions

The major contribution of this thesis is summarized as follows

1. A novel rotary cross-coupled control algorithm has been developed for precision control of a rotary biaxial motion system. The algorithm takes the speciality of the rotary biaxial motion system into consideration to improve the motion performance. Simulation and experimental results have shown that the proposed method has better performance as compared to both the conventional and the direct approximation approaches.
2. Conventional clock synchronization method has been shown unable to generate continuous synchronized clock. An integrated continuous clock state and clock rate synchronization method has been developed. The proposed method overcomes this limitation and generates continuous clock by performing clock state adjustment indirectly through clock rate adjustment. This method is then integrated with clock rate synchronization to achieve better precision. Analysis has shown that the proposed method generates continuous clock with improved clock precision. Experiments have been conducted and verified the analysis.

3. A CAN based network for synchronous motion control has been proposed and implemented. An application layer network protocol has been developed to provide reliable data transmission. It also synchronizes multiple motion nodes through the clock initialization and synchronization algorithm.
4. A prototype of the network based motion control system for fountain application has been built. It consists of the rotary biaxial motion systems, the central controller and the network system. Greater flexibility and variety of fountain design can be achieved by using this system. Higher fountain reliability and better management have been achieved by incorporating the network system.

1.4 Thesis Organization

The rest of the thesis is organized as follows:

In Chapter 2, a brief review in multi-axis motion control is presented. Performance measures of a multi-axis motion control system are discussed. Different control approaches including the conventional axial control approach, the contour based cross-coupled control (CCC) approach and the trajectory coordinate based CCC approach are introduced.

In Chapter 3, a rotary biaxial cross-coupled control algorithm is proposed. The speciality of a rotary biaxial motion system is analyzed and compared to the linear biaxial motion system. The rotary biaxial cross-coupled control algorithm is then proposed based on the unique characteristics. Simulation and experimental results are presented to verify the performance of the proposed algorithm.

In Chapter 4, the CAN based network system for synchronous motion control is presented. Basic concepts in the industrial networks and CAN are introduced. The

network structure and the application layer network protocol are proposed. Three parts of the proposed network protocol, namely the message encoding, decoding and scheduling, the motion data streaming protocol and the motion synchronization protocols are presented in the chapter.

In Chapter 5, an integrated continuous clock state and clock rate synchronization method is proposed. In this chapter, two existing clock adjustment methods, namely the instantaneous adjustment method and the amortization based adjustment method are reviewed. Their limitations in the continuity are discussed. A continuous clock synchronization method is then proposed by adjusting clock state indirectly through clock rate adjustment. This method is then integrated with the clock rate synchronization to further improve the clock precision. Analysis of the proposed method has shown that the synchronized clock is continuous, and the clock precision is improved. Experiments have been conducted and verified the analysis.

In Chapter 6, the prototyping and implementation issues of the prototype are discussed. Three main parts of the system prototype, including the biaxial motion mechanism, the local motion controller and the central controller are described in detail.

Lastly in Chapter 7, a summary of this thesis is presented. Recommendations for future works are also suggested.

Chapter 2

A Review of Multi-axis Motion Control

2.1 Introduction

Multi-axis motion system refers to a system with two or more “axis-of-motion”. The axes could be a linear axis with translational motion or a rotary axis with rotational motion. Multi-axis motion system can be classified as linear, rotary and integrated systems based on the types of the motion axis. For linear multi-axis system, the axes-of-motion are all linear, e.g. an x-y table. On the other hand, all motions axes are rotary type in a rotary system.

Multi-axis motion control aims to control and coordinate each axis to achieve the desired overall performance. In this research, it is one of the main objectives to design a suitable motion controller for the proposed biaxial motion nozzle. To have a good understanding of multi-axis motion control, a review on this topic is presented in this chapter. In section 2.2, the performance measures of multi-axis motion system are introduced. Different control approaches of multi-axis motion system are introduced in section 2.3. The conventional control approach, contour based cross-coupled control approach, trajectory coordinate based cross-coupled control approach and the integrated cross-coupled control approach are discussed in section 2.4, 2.5, 2.6 and 2.7 respectively. Section 2.8 concludes this chapter.

2.2 Performance Measures of Multi-axis motion system

Different performance measures can be used for evaluating the performance of a multi-axis motion system. Typical measures are the tracking error, contouring error and lagging error.

The tracking error e_i of each axis is defined as the axial position difference between the desired and actual trajectories. Denoting the desired trajectory of each individual axis as $r_{id}(t)$ and the actual trajectory as $r_{ia}(t)$, the tracking error e_i can be expressed as:

$$e_i = r_{id}(t) - r_{ia}(t), \quad i = 1, 2, 3, \dots, N \text{ for an } N\text{-axis system} \quad (2.1)$$

The overall tracking error e is defined as the vector difference between the actual position and the desired position. It is the vector sum of all the axial tracking errors expressed as:

$$\bar{e} = \sum_{i=1}^N \bar{e}_i \quad (2.2)$$

The contouring error [1] ε_c is defined as the distance from the actual position to the nearest point on the desired trajectory. It is also called the deviation error [2] since it measures the deviation of the actual trajectory from the desired trajectory.

The lagging error ε_l is defined as the distance between the desired points and the closest point to the actual point on the trajectory. The vector sum of the contouring and lagging errors equals to the overall tracking error, i.e. $\bar{\varepsilon}_c + \bar{\varepsilon}_l = \bar{e} = \bar{\varepsilon}_x + \bar{\varepsilon}_y$.

Fig. 2-1 (a) Performance measures (b) The trajectory coordinate

Fig. 2-1a illustrates the axial tracking error ϵ_x and ϵ_y , the overall tracking error e , the contouring error ϵ_c , and the lagging error ϵ_l in a linear biaxial motion system. In the figure, p_a and p_d refer to the actual and desired positions respectively.

All these performance measures are defined in the reference coordinates, e.g. the Cartesian coordinate. Other coordinates such as the trajectory coordinate is often used in multi-axis system control. As shown in Fig. 2-1b, a trajectory coordinate (n, t) is defined by setting the normal and tangential directions of the desired trajectory as the coordinate axes: n axis and t axis [3]. Hence, the axial errors at n and t axes are defined as the normal error ϵ_n and tangential error ϵ_t respectively. They approximate the contouring and lagging errors respectively provided that the tracking errors are much smaller than the contour radius.

When designing control algorithm for a multi-axis system, the performance measures that may be used depend on the application requirements. For applications such as the cutting machine, the contouring and lagging errors are more important than the tracking errors.

2.3 Control Approaches of Multi-axis System

Many control approaches [4-20] have been proposed and used for multi-axis motion system. In general, they can be classified into two categories: axial control approach and coupled control approach.

Axial control is the conventional control approach in multi-axis motion control. In this approach, feedback control is applied on each axis individually with the control effort focused on improving the accuracy of individual axis [4]. The axial tracking errors are the only performance measures used in the controller. Other performance measures, for example the contouring error, are only affected indirectly through changes of the tracking errors. The contouring error was studied in [5] and concluded that contouring performance can be improved by applying axial control effort to make the dynamic characteristic similar to each other. In addition to feedback control, many other control techniques can be used to control the individual axis. For example, feed forward control can be incorporated into the axial control [4], which further reduce the tracking errors and indirectly the contouring error.

Better contouring performance is achieved by using the coupled approach by considering the coupling effects among axes. There are two approaches of coupling control, namely the “master-slave” approach and the “equal status” approach. In the master-slave approach [6], the results from the master axis serves as one of the inputs to the slave system. Consequently, the master system will affect the slave system greatly, while the slave system will not affect the master system. This approach is used when the dynamics are significantly different among axes. The equal status approach, on the other hand, treats each axis with equal status. Any changes in one axis will affect the other axes. This approach is desirable when the axes have similar dynamics.

Cross-coupled control (CCC) is an “equal status” approach and is one of the commonly used methods. In this method, coupled performance measures like contouring and lagging errors are incorporated into the controller design.

In the following sections, three multi-axis motion control approaches namely the conventional multi-axis control method, the contour based cross-coupled control approach and the trajectory coordinate based cross-coupled control approach are introduced.

2.4 Conventional Multi-Axis Control

Conventional multi-axis motion control method treats each axis as an independent system and controls them separately. Therefore, it can be simplified to a collection of single axis motion control. Fig. 2-2 shows a typical block diagram of a conventional biaxial motion control scheme with a simplified model.

Fig. 2-2 Conventional biaxial motion control

In controlling each individual servo motor, the most commonly used method is the cascaded control approach [7]. A typical cascaded motion control system consists of three control loops namely the torque, speed and position control loops as shown in Fig. 2-3. In the figure, θ_d , ω_d and τ_d are the desired position, speed and torque respectively, while θ_a , ω_a and τ_a are the actual position, speed and torque respectively.

Fig. 2-3 A typical cascaded motion control

To further improve the accuracy of each axis, various control methods can be integrated with the cascaded control scheme. In [8, 9], friction compensation and disturbance observers are added into the feedback loops. Feed forward controller can also be added in addition to the feedback control loops [4].

Besides the cascaded control approach, there are other approaches used for precision motion control, e.g. state space control approach and model predictive control approach [10]. However, these are not the focus of this thesis.

2.5 Contour Based Cross-coupled Control

The basic idea of the contour based cross-coupled control (CCC) method is to estimate the contouring error based on the tracking errors and the contour radius [1]. A contour feedback controller is then used to minimize this error. Further improvement can be attained by introducing variable gains into the contour controller [11, 12], instead of using a fixed control gain. The variable gains are defined based on the estimated contouring error. Other variations and enhancement [13-17] of the method are proposed by integrating the CCC with other control techniques, e.g. feed forward control.

For a linear type multi-axis motion system, the axes are normally designed to be perpendicular to each other. Hence, the motion axes correspond to the Cartesian xyz coordinate axes. In this system, the contouring error can be calculated from the

tracking errors through geometrical analysis. Fig. 2-4 shows a biaxial motion system with motion on x and y axes of a Cartesian coordinate.

Fig. 2-4 Contouring error of a linear biaxial motion system

In Fig. 2-4, the tracking and contouring errors for a desired contour are illustrated. In the figure, (x_a, y_a) is the actual position, (x_o, y_o) denotes the center of curvature of the desired contour, (x_d, y_d) denotes the desired position, e_x and e_y are the x and y axis tracking errors, R is the radius of the contour, α is the transversal angle of the contour, and ε is the contouring error.

In variable gain cross-coupled control, the desired contour is approximated by a circular contour. The contouring error can be estimated as the difference between the circular radius and distance from actual position to the center of the circular contour:

$$\varepsilon = \sqrt{(x_a - x_o)^2 + (y_a - y_o)^2} - R \quad (2.3)$$

where (x_a, y_a) is the actual position which can be represented as

$$\begin{cases} x_a = R \sin \alpha + x_o - e_x \\ y_a = -R \cos \alpha + y_o - e_y \end{cases} \quad (2.4)$$

Combining (2.3) and (2.4) yields

$$\varepsilon = \sqrt{(R \sin \alpha - e_x)^2 + (R \cos \alpha + e_y)^2} - R \quad (2.5)$$

The contouring error in (2.5) can be approximated by Taylor's expansion as

$$\varepsilon = (\cos \alpha + \frac{e_y}{2R})e_y - (\sin \alpha - \frac{e_x}{2R})e_x \quad (2.6)$$

From (2.6), the cross-coupling gains C_x and C_y can be obtained as:

$$\begin{cases} C_x = \sin \alpha - \frac{e_x}{2R} \\ C_y = \cos \alpha + \frac{e_y}{2R} \end{cases} \quad (2.7)$$

The contouring error is hence represented as a function of the axial tracking errors and cross-coupling gains

$$\varepsilon = -e_x C_x + e_y C_y \quad (2.8)$$

If the contour is linear, R is infinite and the corresponding cross-coupling gains becomes

$$\begin{cases} C_x = \sin \alpha \\ C_y = \cos \alpha \end{cases} \quad (2.9)$$

Using (2.7), a variable gain CCC can be constructed by applying a contour controller G_c to the contouring error. The block diagram of this control scheme is shown in Fig. 2-5. In the figure, G_x and G_y are the axial feedback controller, G_c is the contour controller. P_x and P_y are the axial motion system models.

Fig. 2-5 Contour based cross-coupled control for biaxial motion system

2.6 Trajectory Coordinate Based Cross-Coupled Control

The contour based cross-coupled method greatly reduces the contouring error. However, it results in relatively large lagging error, as only the contouring error is included for cross-coupling. This method is thus limited to applications with lower lagging error requirement. To overcome this limitation, the cross-coupled control has been further expanded by considering both the contouring and lagging errors [3, 18, 19]. In [3], a trajectory coordinate is used by defining the normal, binormal and tangential directions as the coordinate axes. The controllers are then designed based on the corresponding axial errors to minimize both the contouring and lagging errors. In the following sections, a brief introduction on the trajectory coordinate based CCC is presented.

In a 2D space as shown in Fig. 2-1b, a trajectory coordinate is defined by setting the normal and tangential directions of the desired trajectory as the coordinate axes. The corresponding axial errors are normal error ε_n and tangential error ε_t respectively. By geometrical analysis, the relationship of the trajectory coordinate and the reference coordinate can be expressed as:

$$\begin{cases} n = -x \sin \alpha + y \cos \alpha \\ t = x \cos \alpha + y \sin \alpha \end{cases} \quad (2.10)$$

where α is the transversal angle. In the trajectory coordinate, the desired position is always at (0, 0). Therefore, the normal and tangential errors can be calculated as:

$$\begin{cases} \varepsilon_n = n_d - n_a = -e_x \sin \alpha + e_y \cos \alpha \\ \varepsilon_t = t_d - t_a = e_x \cos \alpha + e_y \sin \alpha \end{cases} \quad (2.11)$$

The normal error approximates the contouring error by neglecting the curvature of the trajectory. This approximation is valid under the condition that the errors are much smaller than the radius of the curve. From (2.6), the contouring error can be approximated as

$$\begin{aligned} \varepsilon_c &= (\cos \alpha + \frac{e_y}{2R})e_y - (\sin \alpha - \frac{e_x}{2R})e_x \\ &\approx e_y \cos \alpha - e_x \sin \alpha \quad (\text{if } e_x, e_y \ll R) \\ &\approx \varepsilon_n \end{aligned} \quad (2.12)$$

Similarly, the tangential error approximates the lagging error under the condition that the errors are much smaller than the radius of the curvature.

Based on (2.10), a transformation matrix $\mathbf{T}$ that transform the points from the reference to the trajectory coordinate can be defined:

$$\mathbf{p}' = \begin{bmatrix} n \\ t \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{T} \mathbf{p} \quad (2.13)$$

where

$$\mathbf{T} = \begin{bmatrix} -\sin \alpha & \cos \alpha \\ \cos \alpha & \sin \alpha \end{bmatrix}$$

The corresponding axial errors in the reference coordinate can be transformed to the normal and tangential errors as follows:

$$\vec{\varepsilon} = \begin{bmatrix} \varepsilon_n \\ \varepsilon_t \end{bmatrix} = \mathbf{T} \begin{bmatrix} e_x \\ e_y \end{bmatrix} = \mathbf{T}\mathbf{E} \quad (2.14)$$

The feedback controllers $\mathbf{G}_c$ can be designed in the trajectory coordinate space to minimize the trajectory coordinate errors. $\mathbf{G}_c$ is a vector that includes g_n for normal axis and g_t for tangential axis:

$$\mathbf{G}_c = \begin{bmatrix} g_n \\ g_t \end{bmatrix} \quad (2.15)$$

The generated control signals $\mathbf{U}_\varepsilon$ are in the trajectory coordinate. They need to be transformed back to the reference coordinate space to drive the individual axis. The inverse of the transformation matrix $\mathbf{T}$ is applied to $\mathbf{U}_\varepsilon$, in order to get the axial control signals $\mathbf{U}_e$:

$$\mathbf{U}_e = \mathbf{T}^{-1}\mathbf{U}_\varepsilon \quad (2.16)$$

A block diagram of the trajectory coordinate based CCC is shown in Fig. 2-6. As shown in the figure, the controller $\mathbf{G}_c$ is applied in the trajectory coordinate space to minimize both normal and tangential errors.

Fig. 2-6 Trajectory coordinate based cross-coupled control

2.7 Integrated Cross-Coupled Control Approach

Integrated CCC combines the other control algorithms with the CCC, as compared to the PID feedback control based CCC approach. Different integration approaches have been proposed [18-28].

Feed forward control is one of basic control approach. It can improve the tracking accuracy of each individual axis, hence indirectly improve contouring accuracy. By integrating it with feedback control and CCC [18-21] and taking the benefits of feed back control, feed forward control and CCC, both tracking accuracy and contouring accuracy can be improved.

Adaptive control can also be integrated with CCC. By integrating adaptive control into CCC, the controller can be more adaptive to the various changes which may change the system model. In [22], a Kalman filter is applied to recognize the parameters of the system model in real time to cope with the change of load. In [23] a weighted d-step-ahead self tunings adaptive control is combined with CCC to handle the non-linear behavior due to the Coulomb friction.

Optimal [24, 25], Fuzzy logic[26, 27] and Robust control [28] methods have also been integrated with CCC. Better contouring accuracy can be achieved without changing the tracking accuracy. However, there is significant increase in computation, especially if dynamic compensation is used.

A properly designed integrated CCC will generally improve the performance of the multi-axis system in certain aspect. However, the corresponding complexity in controller design and computation of such an algorithm is also greatly increased.

2.8 Conclusions

In this chapter, the topic on multi-axis motion control is reviewed. The performance measures of multi-axis motion system are introduced. Three control approaches namely the conventional control approach, the contour based CCC approach, the trajectory coordinate based CCC and the integrated CCC approach have been discussed. In this research, the rotary biaxial motion system studied has different dynamics as compared to the linear systems. The control of rotary biaxial motion system will be discussed in the next chapter.

Chapter 3

Rotary Biaxial Cross-coupled Control

3.1 Rotary Biaxial Motion System

As introduced in chapter 1, the biaxial motion fountain nozzle is formed by placing the nozzle at the end-effector of a biaxial motion system. This biaxial motion system is rotary based with the two axes placed perpendicular to each other. As a result, the end effector has a spherical motion plane with a fixed radius ρ . It is thus more convenient to define the task space in the spherical coordinate. The two motion axes can be represented by the axes (θ, ϕ) of a spherical coordinate with radius ρ . This representation is equivalent to the latitude-longitude coordinate system used for earth positioning, where θ represents the longitude and ϕ represents the latitude.

As shown in Fig. 3-1, the desired trajectory of a biaxial motion system is a curve on the spherical surface and can be represented by a time series of the positions at θ and ϕ axes. Hence, the tracking errors of both axes are also represented as curves on the sphere. The tracking error curves C_θ and C_ϕ of each axis can be calculated from the angular errors e_θ and e_ϕ , which are defined as the angle difference between the desired and actual trajectories. Fig. 3-1 illustrates the error curves on a spherical surface. In the figure, (θ_a, ϕ_a) and (θ_d, ϕ_d) are the actual and desired positions on the sphere respectively, C_θ and C_ϕ are the error curves of the θ and ϕ axes respectively.

Fig. 3-1 Error curves on the spherical surface

The relation of the error angle and error curve depends on the radius of the curve. As shown in Fig. 3-1, C_θ is on the parallel of latitude, while C_ϕ is on the meridian of longitude. Hence, the radius of the C_θ curve is the radius of the parallel of latitude, while the radius of the C_ϕ curve is the radius of the meridian of longitude. The relationship can be expressed as:

$$\begin{cases} C_\theta = R e_\theta \\ C_\phi = \rho e_\phi \end{cases} \quad (3.1)$$

where ρ is the radius of both the sphere and the meridian of longitude. R is the radius of the parallel of latitude and is calculated as

$$R = \rho \cos \phi \quad (3.2)$$

From (3.1) and (3.2), a relationship between the error curves and the two axes can be obtained:

$$\begin{cases} C_\theta = e_\theta \rho \cos(\phi) \\ C_\phi = e_\phi \rho \end{cases} \quad (3.3)$$

From (3.3), it is observed that C_θ is dependent on both θ and ϕ axis positions. Hence, there is a coupled relationship between the two rotary axes on the spherical motion surface. In contrast, axes of a linear multi-axis system are independent of each other. This is one of the fundamental differences between the rotary and linear biaxial motion systems. Since the cross-coupled control methods reviewed in chapter 2 are derived based on the geometry of linear systems, they can not be used for the rotary system.

In this chapter, a cross-coupled control method suitable for the rotary biaxial motion system is proposed. Section 3.2 presents the proposed CCC. The trajectory generation method is discussed in section 3.4. Section 3.5 describes the experimental setup and the system model identification. Simulation and experimental results are presented in sections 3.6 and 3.7 respectively. Section 3.8 concludes this chapter.

3.2 Rotary Biaxial Cross-Coupled Control

To implement CCC for a rotary biaxial motion system, a simple approach is to ignore the differences between the rotary and linear system by applying the existing linear CCC methods. In this way, the rotary axes θ and ϕ are directly used to replace the linear x and y axes in a CCC. We term this approach the *direct approximation method*. It can be shown that this method yields poor performance. To improve the performance, a new CCC is proposed by considering the dynamics of motion on the spherical surface.

For most applications, the error curves are much smaller than the actual motion curves on the spherical surface. Thus, these curves can be approximated as straight line segments. Moreover, the local spherical surface region can be approximated as a planar surface as shown in Fig. 3-2. This approximation is similar to the case that we approximate the curves on earth as straight lines within a limited region. A Cartesian

coordinate (x', y') can be defined for the local approximated planar surface by utilizing the latitude-longitude-grid on the spherical surface. The x' axis of the Cartesian coordinate is defined as the parallel of latitude in the region, while the y' axis is defined as the meridian of longitude.

Fig. 3-2 The approximated planar Cartesian coordinate on the sphere

Let $\mathbf{P}_{planar}$ and $\mathbf{P}_{spherical}$ be the points on the planar Cartesian and spherical coordinate respectively, the transformation from the spherical coordinate to the approximated planar coordinate is then defined as:

$$\mathbf{P}_{planar} = \begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T}_1 \mathbf{P}_{spherical} = \mathbf{T}_1 \begin{bmatrix} \theta \\ \phi \end{bmatrix} \quad (3.4)$$

where $\mathbf{T}_1$ is a planar approximation transformation matrix and is represented as

$$\mathbf{T}_1 = \begin{bmatrix} \rho \cos \phi & 0 \\ 0 & \rho \end{bmatrix}$$

The tracking error $\mathbf{E}_{planar}$ on the approximated planar coordinate can also be calculated based on the spherical coordinate tracking error $\mathbf{E}_{spherical}$.

Let $\mathbf{E}_{planar} = \begin{bmatrix} e_{x'} \\ e_{y'} \end{bmatrix}$ and $\mathbf{E}_{spherical} = \begin{bmatrix} e_\theta \\ e_\phi \end{bmatrix}$, then

$$\mathbf{E}_{planar} = \begin{bmatrix} e_{x'} \\ e_{y'} \end{bmatrix} = \mathbf{T}_1 \mathbf{E}_{spherical} = \mathbf{T}_1 \begin{bmatrix} e_\theta \\ e_\phi \end{bmatrix} \quad (3.5)$$

On the local approximated planar surface, the axes x' and y' are independent of each others. Hence, the cross-coupled control method for linear biaxial motion system can be applied. In this thesis, we use the trajectory coordinate based cross-coupled control method (section 2.6). Let $\mathbf{P}_{trajectory}$ be the point in the trajectory coordinate. Then the approximated planar coordinate is further transformed into the trajectory coordinate:

$$\mathbf{P}_{trajectory} = \begin{bmatrix} n \\ t \end{bmatrix} = \mathbf{T}_2 \begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T}_2 \mathbf{P}_{planar} \quad (3.6)$$

where $\mathbf{T}_2$ is the trajectory coordinate transformation matrix

$$\mathbf{T}_2 = \begin{bmatrix} -\sin \alpha & \cos \alpha \\ \cos \alpha & \sin \alpha \end{bmatrix}$$

The corresponding axial errors ε_n and ε_t in trajectory coordinate can be obtained as:

$$\vec{\varepsilon} = \begin{bmatrix} \varepsilon_n \\ \varepsilon_t \end{bmatrix} = \mathbf{T}_2 \begin{bmatrix} e_{x'} \\ e_{y'} \end{bmatrix} = \mathbf{T}_2 \vec{e} \quad (3.7)$$

A feedback controller $\mathbf{G}_c$ is applied in the trajectory coordinate space

$$\mathbf{G}_c = \begin{bmatrix} g_n \\ g_t \end{bmatrix} \quad (3.8)$$

where g_n and g_t are the controller for the normal and tangential axes respectively.

The control signals $\mathbf{U}_{trajectory}$ generated through the controller $\mathbf{G}_c$ in (3.8) are for n and t axes in the trajectory coordinate. The inverse of the transformation matrix $\mathbf{T}_2$ need to

be applied to transform $\mathbf{U}_{trajectory}$ into control signals $\mathbf{U}_{planar}$ for x' and y' axes in the approximated planar Cartesian coordinate. Thus, we have

$$\mathbf{U}_{planar} = \begin{bmatrix} u_{x'} \\ u_{y'} \end{bmatrix} = \mathbf{T}_2^{-1} \mathbf{U}_{trajectory} = \mathbf{T}_2^{-1} \begin{bmatrix} u_n \\ u_t \end{bmatrix} \quad (3.9)$$

To transform the control signal in the approximated planar Cartesian coordinate back into the spherical coordinate, we have

$$\mathbf{U}_{spherical} = \begin{bmatrix} u_\theta \\ u_\phi \end{bmatrix} = \mathbf{T}_1^{-1} \mathbf{U}_{planar} = \mathbf{T}_1^{-1} \begin{bmatrix} u_{x'} \\ u_{y'} \end{bmatrix} \quad (3.10)$$

The overall system block diagram is shown in Fig. 3-3. From the figure, it is noticed that three coordinate spaces namely the spherical space, approximated planar space and trajectory spaces are involved.

Fig. 3-3 Block diagram of rotary biaxial cross-coupled control

3.3 Comparison between the proposed and direct approximation methods

Comparing the block diagrams of the proposed method (Fig. 3-3) and the conventional trajectory based CCC (Fig. 2-6), it is clear that the main difference is the added transformation $\mathbf{T}_1$ between the spherical and the approximated planar coordinates. As

we have previously introduced, a direct approximation approach uses the conventional trajectory based CCC by treating θ and ϕ axes as x and y axes. In this section, we analyze the fundamental difference between the direct approximation approach and the proposed method.

In the proposed rotary biaxial CCC, the transformations $\mathbf{T}_1$ and $\mathbf{T}_2$ are used to transform the spherical coordinate into the local Cartesian and trajectory coordinates respectively. Combining the two transformations, an overall transformation from spherical coordinate into trajectory coordinate can be obtained:

$$\begin{aligned}\mathbf{T}_{planar} &= \mathbf{T}_1 \times \mathbf{T}_2 \\ &= \begin{bmatrix} \rho \cos \phi & 0 \\ 0 & \rho \end{bmatrix} \times \begin{bmatrix} -\sin \alpha & \cos \alpha \\ \cos \alpha & \sin \alpha \end{bmatrix} \\ &= \begin{bmatrix} -\rho \cos \phi \sin \alpha & \rho \cos \phi \cos \alpha \\ \rho \cos \alpha & \rho \sin \alpha \end{bmatrix}\end{aligned}\quad (3.11)$$

By combining the biaxial controller $\mathbf{G}_c$ and the coordinate transformation matrix

$\mathbf{T}_{planar}$, an overall rotary biaxial motion controller $\mathbf{G}_{planar}$ can be derived

$$\begin{aligned}\mathbf{G}_{planar} &= \mathbf{T}_{planar} \mathbf{G}_c \mathbf{T}_{planar}^{-1} \\ &= \begin{bmatrix} -\rho \cos \phi \sin \alpha & \rho \cos \phi \cos \alpha \\ \rho \cos \alpha & \rho \sin \alpha \end{bmatrix} \times \begin{bmatrix} g_n \\ g_t \end{bmatrix} \times \begin{bmatrix} -\rho \cos \phi \sin \alpha & \rho \cos \phi \cos \alpha \\ \rho \cos \alpha & \rho \sin \alpha \end{bmatrix}^{-1}\end{aligned}\quad (3.12)$$

ρ can be eliminated in (3.12). Thus, the overall controller $\mathbf{G}_{planar}$ can be denoted as

$$\mathbf{G}_{planar} = \mathbf{T}'_{planar} \mathbf{G}_c \mathbf{T}'_{planar}{}^{-1}\quad (3.13)$$

where $\mathbf{T}'$ denotes the effective transformation matrix

$$\mathbf{T}'_{planar} = \begin{bmatrix} -\cos \phi \sin \alpha & \cos \phi \cos \alpha \\ \cos \alpha & \sin \alpha \end{bmatrix}\quad (3.14)$$

Similarly, we can derive the expression of the overall controller $\mathbf{G}_{direct}$ of the direct approximation method. Since direct approximation method does not have the transformation from spherical space into planar space, it is obvious that

$$\mathbf{G}_{direct} = \mathbf{T}_{direct} \mathbf{G}_c \mathbf{T}_{direct}^{-1} \quad (3.15)$$

where

$$\mathbf{T}_{direct} = \begin{bmatrix} -\sin \alpha & \cos \alpha \\ \cos \alpha & \sin \alpha \end{bmatrix} \quad (3.16)$$

By comparing (3.14) and (3.16), it is noticed that the only difference between the two methods is the additional term $\cos \phi$ in the transformation matrix of the proposed method. The two methods are equivalent if $\cos \phi$ is equal to 1 (when ϕ is equal to 0 or π). Otherwise, the performance of the two methods should be different. It is expected that as ϕ varies, the performance difference between the two methods will also vary. Experiments are performed to verify this expectation in section 3.7.3.

3.4 Trajectory Generation

The trajectories of the rotary biaxial system are more difficult to generate as compared to the linear multi-axis system. This is due to the dependency among axes as shown in (3.3). Considering the circular trajectory on the spherical surface as an example, most of the circular trajectories are too complicated to be expressed explicitly. One exception is the parallel of the latitude as shown in Fig. 3-4. Its trajectory can be represented with ϕ axis being a constant while θ axis rotates constantly as follows

$$\begin{cases} \theta = K_2 t \quad rad \\ \phi = K_1 \quad rad \end{cases} \quad (3.17)$$

Fig. 3-4 Illustration of different circular trajectories with the same radius

For other circular trajectories that have the same radius but located differently on the spherical surface, the axial trajectories will be more complicated as both θ and ϕ axes vary to make a circular trajectory. These trajectories are difficult to express in terms of explicit functions. However, they can be generated by rotating the special trajectory (3.17) through a coordinate transformation. This transformation not only applies to circular trajectory, but applies to all other types of trajectories too.

A coordinate rotation is obtained by consecutive rotations about the x , y and z axes with angles d_x , d_y and d_z respectively [29]. Let the original and the corresponding rotated points be $\mathbf{p}(x, y, z)$ and $\mathbf{p}'(x', y', z')$ respectively, then the new coordinates can be represented as

$$\mathbf{p}' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \mathbf{R}_x \mathbf{R}_y \mathbf{R}_z \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{R}_x \mathbf{R}_y \mathbf{R}_z \mathbf{P} \quad (3.18)$$

where $\mathbf{R}_x$, $\mathbf{R}_y$ and $\mathbf{R}_z$ are the rotation matrices for x , y and z axes respectively

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos d_x & \sin d_x \\ 0 & -\sin d_x & \cos d_x \end{bmatrix}, \mathbf{R}_y = \begin{bmatrix} \cos d_y & 0 & -\sin d_y \\ 0 & 1 & 0 \\ \sin d_y & 0 & \cos d_y \end{bmatrix}, \mathbf{R}_z = \begin{bmatrix} \cos d_z & \sin d_z & 0 \\ -\sin d_z & \cos d_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For the system under investigation, the points are defined in a spherical coordinate. Hence, a transformation from the spherical coordinate to Cartesian coordinate is first performed.

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = r \begin{bmatrix} \cos \theta \cos \phi \\ \sin \theta \cos \phi \\ \sin \phi \end{bmatrix} \quad (3.19)$$

Combining (3.18) and (3.19), we have

$$\begin{bmatrix} \cos \theta' \cos \phi' \\ \sin \theta' \cos \phi' \\ \sin \phi' \end{bmatrix} = \mathbf{R}_x \mathbf{R}_y \mathbf{R}_z \begin{bmatrix} \cos \theta \cos \phi \\ \sin \theta \cos \phi \\ \sin \phi \end{bmatrix} \quad (3.20)$$

Given the rotation angles of the axes, the new coordinate can be obtained by using

(3.20). For example, given $d_x = 0, d_y = -\frac{\pi}{2}, d_z = 0$, (3.20) becomes

$$\begin{bmatrix} \cos \theta' \cos \phi' \\ \sin \theta' \cos \phi' \\ \sin \phi' \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos \theta \cos \phi \\ \sin \theta \cos \phi \\ \sin \phi \end{bmatrix} = \begin{bmatrix} \sin \phi \\ \sin \theta \cos \phi \\ -\cos \theta \cos \phi \end{bmatrix} \quad (3.21)$$

Solving (3.21), the trajectories in the transformed coordinate are obtained as:

$$\begin{cases} \theta' = \tan^{-1}(\sin \theta \cot \phi) \\ \phi' = \sin^{-1}(-\cos \theta \cos \phi) \end{cases} \quad (3.22)$$

By using the transformation, the desired trajectory can be generated more easily. In the following, two trajectory examples are discussed.

A. Circular contour

Consider a circular contour on the spherical surface formulated as

$$\begin{cases} \phi = 0.5 & \text{rad} \\ \theta = 0.1\pi(t-1) & \text{rad} \end{cases} \quad (3.23)$$

The 3D view and axial trajectories of this special circular contour are shown in Fig. 3-5 and Fig. 3-6. By rotating the axes about y axis for $-\pi/2$ rad, a general circular trajectory can be obtained by performing the transformation (3.22). The 3D view and axial trajectories of the new contour are shown in Fig. 3-7 and Fig. 3-8.

Fig. 3-5 3D View of the circular contour (before transformation)

Fig. 3-6 The trajectories of the circular contour (before transformation)

Fig. 3-7 3D View of the circular contour (after transformation)

Fig. 3-8 The trajectories of the circular contour (after transformation)

B. Dumbbell shape contour

The dumbbell shape contour is more complicated than the circular contour. A special case of the dumbbell shape trajectory is defined as follows:

$$\phi = \begin{cases} 0.2, & t \leq 1 \\ 0.16(t-1) + 0.2, & 1 < t \leq 6 \\ -0.16(t-6) + 1, & 6 < t \leq 11 \\ 0.16(t-11) + 0.2, & 11 < t \leq 16 \\ -0.16(t-16) + 1, & 16 < t \leq 21 \end{cases} \quad (3.24)$$

$$\theta = 0.1\pi(t-1)$$

The 3D view and axial trajectories of this special case contour are shown in Fig. 3-9 and Fig. 3-10 respectively. By rotating the axes about y axis for $-\pi/2$ rad, a trajectory with the same shape but different location on the sphere can be obtained using the transformation (3.22). The 3d view and axial trajectories of the new contour are shown in Fig. 3-11 and Fig. 3-12 respectively.

Fig. 3-9 3D view of the dumbbell shape contour (before transformation)

Fig. 3-10 The trajectories of the dumbbell shape contour (before transformation)

Fig. 3-11 3D view of the dumbbell shape contour (after transformation)

Fig. 3-12 The trajectories of the dumbbell contour (after transformation)

3.5 Experimental Setup and System Identification

A prototype of the biaxial motion system has been constructed. Its block diagram and photograph are shown in Fig. 3-13 and Fig. 3-14 respectively. The system consists of a biaxial motion mechanism, an interface board, a controller board and a PC. Fig. 3-14 shows that the biaxial motion mechanism consists of two perpendicularly mounted brushless DC motors, each driven by a servo amplifier. The interface board is used for position feedback and D/A conversion. The PC is used to provide the user interface. It communicates to the controller board through RS233 serial communication. The driving motors are controlled based on cascaded control scheme. The current and speed control loops are implemented in the amplifier, while the position control loop is implemented in the controller board. The sampling period of the position loop is set at 5ms. More details of this prototype are presented in chapter 6.

Fig. 3-13 Block diagram of the experimental setup

Fig. 3-14 Photograph of the experimental setup

In this thesis, the cross-coupled control only implements the position control loop of the motors. Speed control loop and current control loop are implemented individually for each motor. For convenience, we identify the model of the motion axis that includes the motor as well as the speed and current control loops. Matlab system identification toolbox is used to identify the models of the two motors. A square wave signal is given to each axis as the input. The corresponding experimental outputs are

recorded and shown in Fig. 3-15 and Fig. 3-16. Using State Space models with N4SID method, the models of the two motion axes are obtained as

$$\begin{aligned} \theta \text{ axis : } & \frac{0.2344z^2 - 0.3112z + 0.08846}{z^3 - 2.278z^2 + 1.791z - 0.5008} \\ \phi \text{ axis : } & \frac{0.3216z^2 - 0.2337z - 0.04608}{z^3 - 1.851z^2 + 1.232z - 0.3396} \end{aligned} \quad (3.25)$$

Using the identified model, a simulated output for the same square wave input can be obtained and are compared with the experimental output in Fig. 3-15 and Fig. 3-16. It is observed that the simulated results agree well with the experimental results.

Fig. 3-15 Time plot of input and output for θ axis system identification

Fig. 3-16 Time plot of input and output for ϕ axis system identification

The step responses of the two axes are also obtained and shown in Fig. 3-17. It is observed that the θ axis response has a longer response time and smaller gain than the ϕ axis. This observation coincides with the fact the θ axis have heavier load than the ϕ axis, since the ϕ axis is mounted on the θ axis.

Fig. 3-17 Step response of the two axis

To further simplify the model, a first order model of each axis can be constructed based on the step response. The simplified first order models are:

$$\begin{cases} G_{\theta}(s) = \frac{0.98}{s(0.02s + 1)} \\ G_{\phi}(s) = \frac{1}{s(0.013s + 1)} \end{cases} \quad (3.26)$$

This simplified model can be used in the system simulation.

3.6 Simulation Results

With the identified models (3.26), simulation is performed to evaluate the performance of the proposed controller. For fair comparison, both the conventional axial control

approach and the proposed method use the same PID controller. The simulation parameters are listed in Table 3-1.

Spherical radius	$\rho=0.1\text{ m}$
Open loop gains	$K_{\theta}=0.98\text{ s}^{-1}, K_{\phi}=1.0\text{ s}^{-1}$
Open loop time constant	$\tau_{\theta}=0.02\text{ s}, \tau_{\phi}=0.013\text{ s}$
PID parameters	$K_P=5, K_I=15, K_D=1.5$

Table 3-1 List of simulation parameters

In the following discussions, three different methods are simulated and their performances are compared. The “conventional method” is simulated by using individual axial PID controller on each of the two axes. The “direct approximation method” applies linear CCC to the system. The “planar approximation method” simulates the proposed method.

Fig. 3-18 Normal errors (simulation)

Fig. 3-19 Tangential errors (simulation)

The circular contour shown in Fig. 3-8 is used as the desired trajectory for the simulation. Fig. 3-18 and Fig. 3-19 show the normal and tangential errors respectively. The corresponding integral absolute errors (IAE) and maximum errors of the three methods are listed in Table 3-2. It is observed that the IAE of the proposed method is smaller than the direct approach and is only 5.7% of the conventional method for the normal error. Moreover, the tangential error of the proposed method is significantly

smaller than the other two methods. Thus, the proposed method performs comparatively well for both normal and tangential accuracies.

	Normal error IAE (mm)	Normal error max. (mm)	Tangential error IAE (mm)	Tangential error max. (mm)
Conventional method	1.98	1.61	0.78	1.02
Direct approximation	0.31	0.27	0.73	0.95
Proposed planar approximation	0.22	0.20	0.13	0.20

Table 3-2 Comparison of simulation errors with different control methods

From the simulation study, the following results are observed:

- With the proposed planar approximation cross-coupled controller, both the tangential and normal errors are greatly reduced, as compared to the conventional and the direct approximation methods.
- With the direct approximation method, the normal error can be significantly reduced as compared to the conventional method. However, the tangential error is increased.

3.7 Experimental Results

Various experiments have been conducted based on the prototype system to verify the performance of the proposed method. Experiments with different control algorithms at different speeds, desired trajectories and controller parameters are performed. In the following sections, these experimental results are presented and the performance of the proposed CCC is compared with the direct approximation and the conventional methods. The effectiveness of the proposed method at different speed and trajectory radius, as well as the effect of adjusting the normal and tangential gain ratio are also discussed.

3.7.1 Performance of Different Control Methods

The circular trajectory in Fig. 3-8 is used in the experiment. The circular trajectory has an average speed of 0.16m/s. The conventional control method, the direct approximation and the proposed planar approximation method are implemented with the same trajectory. The experimental results of the three methods are shown in Fig. 3-20 and Fig. 3-21. The IAE and the maximum errors are presented in Table 3-3. From these results, it is observed that the direct approximation method yields poor performance, especially the tangential error. With the proposed planar approximation method, both the normal and tangential errors are much smaller than the conventional and direct approximation methods. This result agrees with the simulation results in section 3.6.

Fig. 3-20 Normal error of circular contour

Fig. 3-21 Tangential error of circular contour

	Normal error IAE (mm)	Normal error max. (mm)	Tangential error IAE (mm)	Tangential error max. (mm)
Conventional method	2.9265	3.005	0.76349	1.455
Direct approximation	0.84171	1.995	1.2003	2.45
Proposed planar approximation	0.46531	0.905	0.76349	0.795

Table 3-3 Comparison of errors with different control methods

3.7.2 Performance at Different Speeds

Experiments with different motion speeds have also been performed. For both the direct approximation and proposed methods, circular trajectories with the same radius but two different speeds at 0.08m/s and 0.16m/s are used for the experiments. The experimental results are shown in Fig. 3-22 to Fig. 3-25. The IAE and maximum error at different speeds are listed in Table 3-4. In this table, the difference between the errors of the proposed planar approximation method and the direct approximation method are calculated as percentages.

Fig. 3-22 Normal errors at 0.08m/s

Fig. 3-23 Normal errors at 0.16m/s

Fig. 3-24 Tangential errors at 0.08m/s

Fig. 3-25 Tangential errors at 0.16m/s

	Control method	Normal error IAE (mm)	Normal error max. (mm)	Tangential error IAE (mm)	Tangential error max. (mm)
Speed =0.08m/s	Direct approx.	0.62934	0.915	0.76202	0.83
	Planar approx.	0.45027	0.475	0.49316	0.425
	Difference	28.45%	48.09%	35.28%	48.79%
Speed =0.16m/s	Direct approx.	0.84171	1.995	1.2003	2.45
	Planar approx.	0.46531	0.905	0.5006	0.795
	Difference	44.71%	54.64%	58.29%	67.55%

Table 3-4 Comparison of errors at different speeds

From the experimental results, it is observed that the errors become larger at higher motion speed. It is also observed that at both speeds, the proposed planar approximation method have better performance as compared to the direct approximation approach. At higher speed, the difference between the errors of the direct approximation and the proposed planar approximation method are bigger.

3.7.3 Performance at Different Trajectory Radius

To verify the analysis in section 3.2 that the major difference of the proposed and direct approximation methods is the added $\cos\phi$ term, two circular trajectories with the same speed but different radius are used for the experiment. Thus, the ranges of ϕ axis trajectory of the two profiles are different. Profile 1 and 2 have radius of 84.1mm and 56.5mm respectively. Fig. 3-26 shows the desired ϕ axis trajectory of the two profiles. It is noticed that in profile 2, the ϕ axis trajectory is much nearer to 0 rad as compared to profile 1. Correspondingly, the trajectory of $\cos\phi$ shown in Fig. 3-27 is much nearer to 1 in profile 2 as compared to profile 1.

Fig. 3-26 ϕ axis trajectory of the profiles

Fig. 3-27 $\cos \phi$ trajectory of the profiles

Experiments with the proposed planar approximation method and the direct approximation are conducted with the two profiles. The experimental results are shown in Fig. 3-28 to Fig. 3-31. The corresponding maximum errors and IAE errors with different profiles are listed in Table 3-5.

Fig. 3-28 Normal errors with profile 1

Fig. 3-29 Normal errors with profile 2

Fig. 3-30 Tangential errors with profile 1

Fig. 3-31 Tangential errors with profile 2

		Normal error IAE (mm)	Normal error max. (mm)	Tangential error IAE (mm)	Tangential error IAE (mm)
Profile 1 (84.1mm)	Direct approx.	0.84171	1.995	1.2003	2.45
	Planar approx.	0.46531	0.905	0.5006	0.795
	Difference	44.72%	54.64%	58.29%	67.55%
Profile 2 (56.5mm)	Direct approx.	0.564	1.24	0.55945	1.12
	Planar approx.	0.5311	1.015	0.43427	0.945
	Difference	5.83%	18.14%	22.38%	15.63%

Table 3-5 Comparison of errors with different profiles

From the profile 1 results, it is observed that the errors of proposed method are around 50% smaller than the direct approach. For profile 2, the errors of the proposed method are around 15%. These results agree with the analysis in section 3.3, which shows that the difference between the proposed method and the direct approximation method is the added term $\cos \phi$. Consequently, the improvement becomes smaller when $\cos \phi$ is near unity and larger when $\cos \phi$ is near zero.

3.7.4 Performances at Different Normal to Tangential Gain Ratios

One benefit of the trajectory coordinate based CCC (section 2.6) over the contour based CCC (section 2.5) is the flexibility to trade off between the normal and tangential accuracies. Contour based CCC uses only normal (contouring) error for cross coupling. It is thus unable to improve the tangential accuracy. On the other hand, the trajectory coordinate based CCC uses both the normal and tangential errors for cross coupling. Thus, the tangential and normal gains can be adjusted to achieve different tangential and normal accuracies. Depending on the application’s requirement, one can chose to have a higher tangential accuracy at the cost of the normal accuracy by using a larger tangential gain and smaller normal gain in the cross-coupling controller. Conversely, higher normal accuracy can be achieved by using smaller tangential gain and larger normal gain in the controller.

Three sets of experiments are performed using the proposed CCC with the same circular trajectory, but with different tangential and normal gains in the CCC controller. As shown in Table 3-6, the ratios of normal gain to tangential gain are set as 2:1, 1:1 and 1:2 respectively for the experiments. The total gain, i.e. the sum of the tangential gain and normal gain, is the same for all three cases. The average speed of the desired trajectory is 0.08m/s. Fig. 3-32 and Fig. 3-33 show the normal and tangential errors of the experiments respectively. Table 3-6 compares the IAE and maximum errors of the three experiments.

Fig. 3-32 Normal errors with different gains

Fig. 3-33 Tangential errors with different gains

Pn	Pt	Pn : Pt	Normal error IAE (mm)	Normal error max. (mm)	Tangential error IAE (mm)	Tangential error max. (mm)
16	8	2:1	0.379	0.365	0.663	0.60
12	12	1:1	0.506	0.525	0.547	0.485
8	16	1:2	0.629	0.65	0.398	0.355

Table 3-6 Comparison of errors with different tangential to normal gain ratio

From the results, it is observed that the normal accuracy is improved at the cost of tangential accuracy if the normal gain is twice the tangential gain. Conversely, the tangential accuracy is improved at the cost of normal accuracy, if the tangential gain is twice the normal gain. Thus, the tangential and normal accuracies can be traded off by adjusting their gain ratio to meet the application requirements.

3.8 Conclusions

In this chapter, we have analyzed the dynamics of a rotary based biaxial motion system. As its dynamics are different from the linear biaxial motion system, direct application of the cross-coupled control method derived based on linear multi-axis motion system yields poor performance. To overcome this problem, a planar approximation based method is proposed in this thesis. Both simulation and experimental results have verified that the proposed method greatly reduces both the tangential and normal errors.

Chapter 4

Network System Design

4.1 Introduction

One of the objectives of this research is to design and implement a reliable network system for controlling multiple biaxial motion systems. This chapter presents the network system design including the network structure and protocol. In this thesis, the network structure adopts a hierarchical approach. In this structure, the local motion controllers are connected and controlled by a central controller through the network.

A CAN based network protocol is proposed for the network system. The block diagram of the proposed network protocol is shown in Fig. 4-1. The physical and data link layer in Open System Interconnection (OSI) model of the network is defined by CAN, which provides a reliable real time communication protocol. However, the CAN standards do not specify how application data are transmitted in the network. Hence, an application layer network protocol is proposed to meet the data rate, timing constraints and synchronization requirements of motion control. As shown in Fig. 4-1, the application layer network protocol includes three major blocks namely the message encoding, decoding and scheduling protocol, the motion data streaming protocol and the motion synchronization protocol. The message encoding, decoding and scheduling protocol is responsible for decoding the received CAN message, encoding the application data into CAN message, and scheduling the message transmissions based on their priorities. The motion data streaming protocol provides a reliable mechanism

to periodically update motion data for local motion controllers. The motion synchronization protocol manages the local motion controller in order to achieve synchronized motions among different nodes.

Fig. 4-1 Overview of the proposed network protocol

In the following sections, the details of the network system are discussed. In section 4.2, a review on industrial networks including the CAN is presented. The structure of network system is studied in section 4.3. The message encoding, decoding and scheduling protocol is described in section 4.4. In section 4.5, the details of the motion data streaming protocol are presented. Lastly, the motion synchronization protocol is presented in section 4.6.

4.2 Industrial Network and Controller Area Network

4.2.1 Industrial Networks

In the modern industrial and enterprise area, network plays a very important role. It is used to deliver management and financial information, share working information,

perform factory automation and control individual devices. Depending on the application, a network can be categorized into 3 levels.

The highest level is the information level. This is the level that the enterprise management information including financial, inventory, production, sales data etc are delivered. Network at this level is also called information network or data network. A few communication protocols such as Ethernet, Wireless LAN and ATM can be used for information network. At this level, the data exchanged on the network are typically of high volume. Hence, high data rate ranging from 10M to 10G bps is typically required. On the other hand, the timing requirement is relatively low.

The second level is the process level. At this level, the factory/plant automation is implemented. It is used to connect the various controllers together such as PLC or microcontroller through a network. The network in this level is also called process network. Typically, the communication protocols used in process network are field buses due to the high timing requirement.

The third level is the device level. It is used to connect low level devices to the controller. At this level, the connection can be point to point or through a device-level network, depending on the number of the devices connected. Normally, if the number of devices is relatively small, it is desirable to use point-to-point connection. If the number of connected devices is large, it is better to connect them through a network to avoid complicated wiring and provide flexibility for expansion. The corresponding network protocols are typically field buses.

Since process networks and device network are mostly used in industries, they are also called industrial networks. Industrial networks are special LANs that are dedicated to data acquisition and the control of sensors and actuators. Most of the industrial

networks are deterministic and are more suitable for real time applications. However, they have lower data rate as compared to the Ethernet, typically from few Kbps to few Mbps. In comparison, Ethernet has a very high speed, from 10Mbps and upto 10Gbps. However, it is not deterministic [30] and cannot be directly used for real time application. Recently, there are also research works trying to use the Ethernet as real time control network [30].

Numerous industrial network protocols have been proposed. Some common industrial network protocols are CANopen, Device-Net, Profibus, P-net, Mod-Bus, Inter-Bus, LonWorks, real time Ethernet, etc. These protocols have their own characteristics and are designed for different applications. Despite the numerous industrial network protocols, many are based on the same physical and/or data link layer using the same communication standard. The most widely used communication standards are RS232, RS422/485, CAN and Ethernet. Many proprietary protocols are developed based on these standards. For example, RS485 standard is adopted in Profibus, P-net, Mod-Bus and Inter-Bus, while CAN is the basis for Device-Net, CANopen and Safety-Net. The use of CAN based protocols have increased significantly in recent years, due to its high data efficiency and reliability as compared to the older RS485 standard. More details of CAN is introduced in the next section.

4.2.2 Controller Area Network

In this thesis, the network system is based on CAN. The CAN Network is a serial communication protocol initially developed by BOSCH [31]. It later becomes an ISO standard (ISO 11898 and ISO 11519) [32, 33]. It is designed to operate at upto 1Mbps data rate. It's capabilities of error detection, error confinement, and collision arbitration makes it suitable communication bus for real time applications.

CAN is initially designed to provide a reliable, efficient and cost effective communication solution for automobiles. It is used to replace the wires connecting the various automobile electronic control systems such as anti-lock braking system, engine management system, traction control, etc. Great improvements in reliability and cost-effectiveness have been achieved by utilizing the network. The usage of CAN later extended into other industrial areas.

In the following sections, the CAN standard including its physical layer, data format and fault detection mechanism is discussed.

4.2.2.1 Physical Layer

The CAN standard specifies the physical layer and data link layer of the OSI network model. The physical layer of CAN specify the driver/receiver characteristics, bit encoding and bit timing synchronization.

As specified in ISO11898, CAN bit stream is coded according to the Non-Return-to-zero (NRZ) method. Two logical states, recessive and dominant states are specified. Dominant states overwrite the recessive when collision happens for arbitration.

The exact transmission media is not specified in CAN. Thus, different media including electrical, optical, power-line and wireless can all be used as CAN transmission media. The most often used media is electrical with differential signaling. Since CAN adopts a bus topology, the two ends of the bus must be terminated with 120Ω terminal resistance to avoid signal reflection. Repeaters, bridges or gateways can be used to extend the physical limit.

On the physical layer, CAN uses synchronous bit transmission to enhance the transmission capacity. Synchronization is required when any node receives a data

frame or remote frame. Since there is no explicit clock signal for reference in CAN, two mechanisms, namely hard synchronization and resynchronization are used to maintain synchronous transmission. Hard synchronization is performed at the “Start of Frame”, while resynchronization is performed for the subsequent bits if necessary. In CAN, one bit time is divided into a few segments. When it is detected that bit edge does not happen in the desired bit time segment, resynchronization is performed.

4.2.2.2 Data Format and Arbitration

The CAN message is transmitted in frames. Each frame consists of 7 fields, including Start of Frame (SOF), Identifier field, Data field, Cyclic Redundancy Code (CRC) field, ACK and End of Frame (EOF) as shown in Fig. 4-2. There are two types of frames, standard frame and extended frame. The difference is mainly on the length of the identifier. Standard frame contains 11 bit identifier, while extended frame contains 29 bit identifier. Increase in the identifier length increases the number of unique identifiers, but also reduces the data transmission efficiency, since the overhead is also increased. The data field of a CAN frame contains data to be transmitted. The maximum number of data contained in the data field is 8 bytes. Control field includes flags indicating the format and data length. CRC field contains 15 bit redundancy check code.

SOF	Identifier	Control Field	Data Field	CRC Field	ACK	EOF
-----	------------	---------------	------------	-----------	-----	-----

Fig. 4-2 CAN message frame format

One of the unique characteristic of CAN is its non-destructive bitwise arbitration. It is an enhanced capability added to the Carrier Sense Multiple Access with Collision Detection (CSMA/CD). CAN uses CSMA/CD for media access control and collision detection. However, unlike the conventional CSMA/CD, where both the collided

frames are discarded at collision, only one frame is discarded in CAN with arbitration. The arbitration is based on the priority of frame which is determined by the identifier field. The frame with higher priority survives after collision, whereas the lower priority one is discarded. In electrical media logical zero is the dominate state. Thus, smaller identifier number corresponds to higher priority.

Non-destructive bitwise arbitration gives the flexibility to allocate the bus based on need and priority. It also results in deterministic message delay. In CAN, message delay may include the preparation time, the transmission time and the arbitration and queuing time [34]. The first two delay times are roughly constant for a given CAN network. The only variable time delay is then the arbitration and queuing time. However, this time delay is dependent on network load and the message priority. The higher priority, the shorter arbitration and queuing delay can be achieved. Therefore, by assigning different priority, different delays levels can be controlled. If a message is assigned the highest priority, then there's no arbitration and queuing delay.

4.2.2.3 Error Detection and Error Confinement

CAN also has strong error detection capability. Five error detection mechanisms including three at message level and two at bit level are defined. These error detection mechanisms result in very low bit error probability.

At the message level, the three error detection mechanisms are the CRC, Frame Check and Acknowledgement Error Check. In every CAN frame, there is a 15 bit CRC field. The CRC is computed by the transmitter based on the message content. Upon receiving the message frame, the receiver calculates and compares its CRC with the received CRC to detect message error. Frame Check mechanism checks certain predefined bit value in the frame at certain points. If an invalid bit is found, form error

is detected. Acknowledgement Error Check reports an error if a transmitted message has not been acknowledged.

At the bit level, the bit monitoring and bit stuffing mechanism are implemented. The bit monitoring mechanism of a transmitter automatically monitors the bus and compares the transmitted bit level with the actual bit level. Bit stuffing is a mechanism that adds an additional bit with opposite polarity after 5 consecutive identical bits. Hence, if 6 or more identical consecutive bits are received, a stuff error is detected.

By combining the five error detection mechanisms, CAN provides a very reliable error detection capability. Analysis [35] has shown that CAN have a better performance in terms of bit error probability compared to other serial bus protocols.

CAN also implements mechanism to determine whether there's temporary error or permanent error. This mechanism is called error confinement. The basic idea of the mechanism is to record the numbers of occurrence of different errors. Different CAN controller states are then determined based on the number.

4.2.2.4 Summary and Network Selection

In summary, CAN have the following main characteristics:

High speed: CAN supports upto 1Mbit/s speed, which is much higher than some other commonly used serial bus standard, e.g. RS485 (125kbit/s).

Cost-effective: The commonly used transmission media is twisted wire pairs, which is much cheaper as compared to the multi-core or coaxial cables used in other bus systems. Furthermore, CAN controllers are widely available either integrated with microcontrollers or as a standalone chip, both at very low cost.

Deterministic time delay: By implementing the non-destructive bitwise arbitration, the message time delay is deterministic. Network resources can be flexibly allocated based on need and priority by assigning different message identifiers.

Multi-master and Peer-to-peer communication: Each CAN node can communicate with any other nodes on the network. Hence multi-master communication mode can be configured. Peer-to-peer communication can also be implemented in CAN.

Broadcasting capability: In CAN, each message carries an identifier instead of the destination address. Broadcasting can be easily achieved by configuring one identifier to be accepted by a group of nodes.

Reliability: By implementing the various error detection techniques, very high reliability is achieved in CAN.

In consideration of these characteristics, CAN is a very suitable network standard for network based motion control application. Its high data rate can support the relatively high network traffic load in the application. Its reliability and deterministic property support the reliable transmission of real time motion data. Its broadcasting capability is very useful for controlling multiple slave nodes simultaneously. Therefore, CAN is used as the basis of the network system in this thesis.

4.3 Network System Structure

The structure of the network system depends on its application. In this thesis, the network system is used for a network based control application. Network based control refers to control application that relies on a network to deliver the control command, control input and/or feedbacks. It is used in system with multiple nodes to replace the

traditional point to point wirings. This will greatly reduce the complexity, increase the flexibility and reliability of a control system [36].

There are two main approaches to utilize the network for controlling multiple nodes [37]. The first is the hierarchical approach. In this approach, each process has its own local controller which is physically nearby. These local controllers are connected to a central controller through the network. The central controller sends the command and set-points to the local controllers through the network. It may also receive data from the local controllers, but only for monitoring purpose. Local controllers perform motion control based on the received commands and set-points. In such system, the feedback control of each local node is performed locally. No feedback control loop is closed through the network. An example of a network based control with this approach is illustrated in Fig. 4-3.

Fig. 4-3 Hierarchical approach of network based control

The benefits of this approach is its modularity, as each node can be easily configured. It is scalable and can be used for both large and small scale systems. Since no feedback loop is closed through the network, the network delay and data rate requirements are much lower. However, this approach is only applicable for systems

whose actuators and sensors of the plant are closely located. For system whose sensors and actuators are physically far separated, this approach is not suitable. In the industrial, many large scale control systems such as the power plant use this approach.

The second approach is the distributed approach which is illustrated in Fig. 4-4. In this approach, the network is directly used to deliver the control and feedback signals among the controllers, actuators and sensors. Controllers for each process may be physically located differently from the actuator and/or sensors, but connected through the network. Closed loop control of each process is performed through the network.

Fig. 4-4 Distributed approach of network based control

This approach provides flexibility in locating the controllers, actuators and sensors. Since the sensor data is transmitted through the network, it can be designed to be received by multiple nodes. Thus, it is much easier for interactive control of different processes. However, the network introduces variable delay in both control input and feedback. Consequently, the design of a reliable controller is more difficult.

In this thesis, the actuators (motors) and sensors for each multi-axis motion system are located together. Furthermore, little interaction is required among the local nodes. Considering these requirements, the hierarchical approach is adopted.

The CAN supports multi-master configuration. Thus, multiple central nodes can be used and configured with different functions. For example, one central controller can

be configured for controlling purpose, while another configured for monitoring only. Alternatively, the multiple nodes can be configured as active node and standby nodes. Standby nodes are used as redundant nodes to replace the active node in case it fails. In both examples, the reliability of the system can be improved using the multiple central nodes.

Combining the multi-master and hierarchical approach, the structure of the network system is formed as shown in Fig. 4-5. The proposed network system consists of two central nodes and multiple local nodes. Based on the capability of CAN, a maximum of 100 nodes can be supported. These nodes are connected through a CAN bus. Each local node consists of a local controller and a biaxial motion mechanism. The local controller receives the command from the central nodes and controls the motion of the biaxial motion mechanism.

Fig. 4-5 The proposed network system structure

4.4 Message Encoding, Decoding and Scheduling

In this section, the message encoding, decoding and scheduling protocol is defined for our targeted application. Application data types and CAN messages formats are defined for message encoding and decoding. Message scheduling is achieved by dynamically assigning the CAN identifier of each message. The message encoding and

decoding standard and priority based message scheduling method are presented in sections 4.4.1 and 4.4.2 respectively.

4.4.1 Message Encoding and Decoding

In the network based motion control system, several types of data are transmitted through the network. These data include commands, requests, motion data and feedbacks. Commands are sent by the central controller to control the action of local controllers. Requests are sent by the local controller to demand for certain actions from other nodes. Motion data are the desired trajectories for motion control at the local controllers. Feedbacks are the monitoring data send from local controller to central controller for status recording. These data are classified into different types and have been assigned unique data headers. The detail of data type assignment is shown in Appendix A.

All the application data are encoded into CAN messages based on the data format. Data format is defined in the data field of CAN messages, as illustrated in Fig. 4-6. For each message, the first byte is a header that specifies the data type. The last byte contains the sender’s ID and the rest bytes are application data. According to CAN standard, the data field can contain upto 8 bytes of data. Hence, the application data can be 0 to 6 bytes, while the overall data field length is ranged from 2 to 8 bytes.

CAN data field:

Header	Application data	Sender’s ID
1 byte	0 to 6 bytes	1 byte

Fig. 4-6 Application data format

4.4.2 Priority Based Message Scheduling

In real time networks, messages might have different timing constrains. Some have very stringent timing constrain, while others are less restricted. It is thus important to

handle different timing constraints and guarantee the job deadlines of the messages. This is when message scheduling comes into place. In this section, the priority based message scheduling method is presented.

Classification of network messages can be performed based on their timing constraints. One of the most common timing constraints is the deadline. The deadline of a message refers to the time before which the message transmission has to be completed. Message transmitted after its deadline is useless. In a hard real time application, critical error or failure occurs once the deadline is missed. In soft real time application, a few missed deadlines can be tolerated. While a non-real time application has no timing constraints.

Message scheduling is to schedule and allocate network resources for each message based on their real time requirements and deadlines. There are two approaches in message scheduling. One is the priority based approach, in which messages are assigned different priority based on their timing constraints [38-40]. The other approach is the time based approach that allocates different time slots for different message types [41, 42].

As introduced in section 4.2.2, a contention based collision arbitration mechanism based on the identifiers is used. Different identifiers can be assigned for different message types with different level of priorities in competing for the network resource. This mechanism can be utilized for message scheduling as the identifier implies the priority of the message. A few network scheduling algorithms based on CAN have already been proposed. These include the fixed priority approach [43, 44] and the dynamic priority approach [39, 40]. In the fixed priority approach, the identifier of the messages is assigned off-line based on their priorities. The identifier cannot be

changed online once the network has started operation. In the dynamic priority approach, the identifiers are dynamically assigned based on both the message priority and deadline.

In this thesis, we implement a simple but efficient priority based message scheduling with dynamic priority assignment. In this scheduling method, the identifier of CAN message is divided into three parts, namely the priority, deadline and receiver ID as shown in Fig. 4-7.

Fig. 4-7 CAN identifier assignment

The priority field is assigned off-line based on the real time requirement of the messages. Four levels of real time requirements: critical, hard real time, soft real time and non-real time are defined, with the four corresponding types and priorities of messages. The priority for the critical level is the highest and lowest for the non-real time level, as listed in Table 4-1.

Message type	Priority level (0 is highest)
Critical	0
hard real time	1
soft-real time	2
Non-real time	3

Table 4-1 List of priority levels of message types

For messages with the same priority, the tightness of deadline may still be different. Hence, we introduce an online assigned deadline field in the identifier. The deadline field is represented by the “available time period”, which measures the time interval from the current time to the message deadline. Since the deadline field is only 2 bit long, quantization is performed to distribute the available time period into 4 levels.

The remaining seven bit of the identifier is used for receiver ID. Receiver ID specifies the receiver of the message. It may be single node ID referring to a single receiver node or a group ID referring to a group of receiver nodes. Seven bit receiver ID provides a maximum of 128 unique IDs which is enough for our intended application.

By using the above three field, the identifier of a message is assigned dynamically based the message priority, deadline and the receiver ID. The proposed method is very simple to implement and add no extra traffic load to the network. It can effectively improve the reliability of the network and guarantee the timeliness of different network messages.

4.5 Motion Data Streaming

As introduced in section 4.3, the network structure follows a hierarchical approach. In this system, the desired trajectories for each local controller are delivered from the central node through the network in the form of motion data. They are then used for tracking control by each local controller. The motion data are required to be transmitted to the desired nodes accurately, timely and in sequence. Any transmission error, such as inaccurately transmitted data, wrong data sequence or failed to deliver on time would result in unsatisfactory performance. Hence, it is desirable to have a reliable mechanism to guarantee the transmission requirements are fully met.

In this thesis, we use the motion data streaming protocol to provide reliable data delivery. This protocol is a combination of a series of data transferring and validation methods. These methods include trajectory down-sampling and motion data scheduler at the transmitter side, as well as the motion data sequence control, buffering and trajectory up-sampling at the receiver side.

Fig. 4-8 Motion data streaming mechanism

An overview of the motion data streaming mechanism is shown in Fig. 4-8. As shown in the figure, the desired motion trajectory is first down-sampled, before being transmitted as motion data by the motion data scheduler. At the receiver side, the received motion data must have gone through sequence check before being stored in a buffer. Requests may be sent back to the transmitter's motion data scheduler for data retransmission or transmission rate adjustment. The buffered motion data are then reconstructed back to the desired trajectory through up-sampling. The details of each block of the motion data streaming are introduced in the following sections.

4.5.1 Trajectory Down-Sampling

Delivering the motion data generates heavy load to the network. In this application, the frequency of the periodic motion data transmission on the network is 200Hz. Since upto 100 multi-axis motion systems are controlled simultaneously through the network, the network traffic load would be very high. The load may exceed the capability of CAN. For example, suppose 60 multi-axis motion systems are controlled through the network, and each motion data is 100 bit long including the overhead. Then the traffic load is 1.2Mbps, which exceeds the maximum bit rate of 1Mbps. Hence, we need to down-sample the desired trajectory to reduce the traffic load.

In motion control, high sampling rate are used in the closed loop motion control for high accuracy. This sampling frequency is much higher (>100 times) than the signal frequency of the desired trajectory. Under this condition, the desired trajectory sequence can be reconstructed without significant errors by performing down-sampling first and then up-sampling. By transmitting the down-sampled trajectory instead of the original trajectory, the network traffic load can be greatly reduced several fold.

To perform down-sampling, a down-sampling factor n is first specified. In this application, n is selected to be 5. Down-sampling is then performed by picking out every n^{th} sample, i.e.

$$g(k) = h(mk) \quad (4.1)$$

where $g(k)$ is the down-sampled sequence, $h(k)$ is the original sequence.

The corresponding up-sampling performed at the receiver side to reconstruct the original desired trajectory will be presented in 4.5.5.

4.5.2 Motion Data Scheduler

The motion data scheduler is implemented at the transmitter side. It is responsible for periodically transmitting the motion data based on the down-sampled desired trajectory. The transmission period T_t is equal to the product of the sampling period and the down-sampling factor, i.e.

$$T_t = n \times T_s \quad (4.2)$$

where T_s is the sampling period of the motion control system

The motion scheduler converts the down-sampled desired trajectory into a sequence motion data. Each motion data is one sample of the desired trajectory. A sequence

number is assigned to the motion data based on its actual sequence. In each transmission period, one piece of motion data including the sequence number is transmitted to each of the local nodes.

The motion data scheduler also receives feedback requests from the local nodes. It then adjusts its motion data transmission operation correspondingly for the requested local nodes. A list of feedback types and the corresponding responses are listed in Table 4-2.

Feedback type	Description	Response
Retransmission	Request to retransmit the motion data with a specific sequence number	Send the motion data from the requested sequence number onward
Buffer low	Buffer level low, request to transmit more motion data	Send motion data at double transmission rate for 10 sampling periods
Buffer high	Buffer level high, request to transmit less motion data	Send motion data at half transmission rate for 10 sampling periods
Motion stops	Motion terminated at the local node unexpectedly	Stop sending motion data

Table 4-2 List of feedbacks and Responses

4.5.3 Motion Data Sequence Control

Desired trajectory in the form of motion data must have the correct sequence. Disordered motion data will generate wrong trajectory and may cause serious problem. However, since the motion data of the desired trajectory is transmitted individually from the central node to the local controllers through the network, it is possible that the data arrives in the wrong order or lost due to transmission error. Therefore, it is important to implement a mechanism to validate and maintain the correct sequence of motion data.

In this study, a sequence control mechanism has been implemented to achieve this goal. In this mechanism, the sequence number transmitted together with the motion data is utilized. The receiver maintains an *expected sequence number*, which is the sequence number of the motion data that is expected to be received. This expected sequence number is then used to determine whether a newly received data is the desired data, by comparing the received and expected sequence numbers. Corresponding actions of accepting data, rejecting data or requesting for retransmission are then performed based on the result. The state diagram of the sequence control mechanism is shown in Fig. 4-9.

Fig. 4-9 State diagram of sequence control mechanism

As shown in the figure, the sequence control mechanism defines two receiver states: the normal state and resending state. Initially the receiver is in the normal state. Upon receiving a new motion data, the controller compares the expected and received sequence number to determine if the correct data has been received. If it is correct, it updates the expected sequence number and proceeds to the next stage of the motion data streaming protocol. If the received sequence number is bigger than the expected sequence number, it discards the data. If the received sequence number is smaller than

the expected sequence number, the mechanism enters the resending state and request for retransmission of the data with the expected sequence number.

In the resending state, the receiver waits for the motion data with the expected sequence number. All data with unexpected sequence numbers are discarded. Upon receiving the expected sequence number, the receiver goes back to the normal state. Furthermore, the receiver in the resending state keeps a retransmission counter C . Counter C is increased by one at each receiving instance in the resending state. If the C is greater than a pre-selected threshold number e.g. 4, the receiver assumes that the previous retransmission request has not been successfully delivered and it sends a retransmission request message again. This is to ensure the effectiveness of the mechanism in case of message lost. The pre-selected threshold number depends on the maximum network delay.

4.5.4 Motion Data Buffering

In the targeted application, motion data for upto 100 nodes are required to be sent to the local controllers simultaneously every 25ms. Hence, there is a heavy traffic of real time message being transmitted on the network. Though network has been properly scheduled as discussed in section 4.4.2, it is difficult to guarantee the correct and timely transmission of every motion data.

To overcome this limitation, buffering of the motion data is performed. A circular buffer is used to temporally store the received motion data before it is being used for motion control. By using the buffer, a waiting time is created. It is defined as the time interval between a motion data being received and used for motion control. The waiting time provides a longer time interval for sequence control and retransmission to guarantee the correctness of motion data transmission. The length of waiting time is

equal to the product of buffer level and sampling period. The buffer size can be determined based on the network condition. With larger buffer size, it is more flexible for motion data transmission and management due to the longer waiting time.

Buffer level refers to the number of valid motion data stored in the buffer. The buffer should be maintained at a desired level to avoid buffer empty or over flow. The ideal buffer level is around half the buffer size. Actual buffer level depends on the rates of receiving and consuming of the motion data. It may vary due to the mismatch between the central and local nodes' actual sampling rates. A buffer management mechanism is thus implemented to dynamically manage the buffer by adjusting the transmission rate.

To manage the buffer, two indication levels, namely the buffer low indication and buffer high indication are set. A diagram illustrating the buffer stack and the indication levels is shown in Fig. 4-10. As shown in the figure, Buffer low indication is set at a few data above the starting level of the buffer stack, while buffer high indication is set at a few data below the full level. If the buffer level reaches the buffer low indication level, a feedback message will be sent to the central node to request for a faster transmission rate. Conversely, if the high buffer indication level is reached, a request message is sent to request for slower transmission rate. The central node will change the transmission rate correspondingly until the buffer level has returned back to a pre-defined level. In this way, the buffer can be maintained within a range, avoiding buffer empty or over flow.

Fig. 4-10 Buffer stack and indication levels

In this study, the buffer size is set at 30. The buffer low indication is set at 13, while the buffer high indication is set at 25. An experimental result of the buffer management is obtained from the implemented prototype. In this experiment, the sampling period of the central node is 1% smaller than the sampling period of the local node. The buffer level of the local node is recorded and shown in Fig. 4-11. In this figure, it can be observed that the buffer level is maintained around half to two third of the maximum buffer size. The buffer level tends to drop slowly, due to the mismatch of the central and local nodes' sampling rates. When the buffer level drops below 13, it is increased rapidly due to the doubled data transmission rate from the central controller. The transmission rate is then set back to normal after the buffer level is raised.

Fig. 4-11 Experimental result of the managed buffer

4.5.5 Trajectory Up-Sampling

Up-sampling are performed at the receiver side to reconstruct the original trajectory by using the motion data stored in the buffer. Desired trajectory up-sampling is achieved in two steps: first reconstruct the continuous time trajectory through interpolation and then sample the reconstructed trajectory at the desired sampling rate.

In this application, the second order polynomial interpolation is adopted for trajectory reconstruction. Polynomial interpolation can generate much smoother trajectories than the linear interpolation method. In the second order polynomial interpolation, three points are used to interpolate the trajectory in between two adjacent discrete trajectory points. The curve constructed by the three points is represented by a second order curve model as:

$$At^2 + Bt + C = x \quad (4.3)$$

where t denotes the time, x denotes the position, A , B , and C are the coefficient of the curve equation. By utilizing the positions of the three points, three equations can be formulated. Given the three points on the curve: $P_1(t_1, x_1)$, $P_2(t_2, x_2)$ and $P_3(t_3, x_3)$, the three equations are

$$\begin{cases} P_1 : At_1^2 + Bt_1 + C = x_1 \\ P_2 : At_2^2 + Bt_2 + C = x_2 \\ P_3 : At_3^2 + Bt_3 + C = x_3 \end{cases} \quad (4.4)$$

Since the up-sampling factor is 5 in this application, the sampling periods of the received and up-sampled trajectories are $5T_s$ and T_s respectively. Hence, t_1 , t_2 and t_3 are equal to 0, $5T_s$ and $10T_s$ respectively. Based on (4.4), the curve coefficients can be solved as

$$\begin{cases} A = \frac{x_3 - 2x_2 + x_1}{50T_s^2} \\ B = \frac{4x_2 - x_3 - 3x_1}{10T_s} \\ C = x_1 \end{cases} \quad (4.5)$$

Based on the calculated coefficients of the curve, the up-sampled trajectory samples at sampling instances $t = 0, 1T_s, 2T_s, 3T_s$ and $4T_s$ can be obtained. Trajectory up-sampling is achieved by repeating this process for all the trajectory samples.

4.6 Motion Synchronization

The message scheduling and motion data streaming protocols provide reliable data transmission on the network. Besides that, motion synchronization among multiple nodes is also required. This is achieved through the motion synchronization protocol presented in this section.

Synchronization of motion refers to the requirement that the desired trajectories of different nodes must be processed synchronously. In terms of sequence number, synchronized motion means that the desired position for control at each node must have the same sequence number at any time instance. Therefore, motion

synchronization is essentially to synchronize the *current sequence number* of the motion data being used by various nodes at the current sampling instance.

In the system, the controller output is generated by using the motion data in sequence at each sampling instance. The current sequence number is incremented at each sampling instance. Hence, it resembles a clock that starts from 0 and ends at the last sequence number of the desired motion trajectory. Each sampling instance becomes a clock tick instance. The motion synchronization then becomes a clock synchronization problem.

At each node, the clock (current sequence number) is generated through timer interrupt, which is dependent on the outputs of a crystal oscillator. As the clocks may drift over time, it results in different clock rates among different nodes. This may lead to unsynchronized motion.

In this thesis, we proposed the corresponding clock initialization and clock synchronization methods based on CAN. Based on the broadcasting characteristic of CAN discussed in the section 4.2.2.4, clock initialization can be easily performed. Before starting a motion control session, a “start” command is broadcasted to every local motion controller with the highest priority. Every nodes can receive this command with very small variations. Upon receiving the command, each node resets their clock and starts operation. The initial clocks hence will be initialized with minimum difference.

After the initialization, the clock synchronization must be performed periodically. In motion control, the current sequence number must be monotonically increasing with unit step size. Skipping or repeating of the current sequence number results in poor performance, such as vibration or shaking. Since the clock is the current sequence

number, it is required that the clock should be maintained continuous and monotonically increasing. Conventional clock synchronization methods result in discontinuous clock which is undesirable for motion control. In this thesis, we propose a continuous clock synchronization method that suits well for this application. Detail of the proposed method is presented in the next chapter.

4.7 Conclusions

In this chapter, the design of a network for synchronous motion control system is presented. It is based on CAN standard and adopts a hierarchical structure. Application layer protocol has been designed. It includes the message encoding, decoding & scheduling, the motion data scheduling and the motion synchronization protocols. These protocols ensure reliable data transmission for the motion control application and synchronize the motion among multiple local nodes.

Chapter 5

Continuous Clock State and Clock Rate Synchronization

5.1 Introduction

As introduced in Chapter 4.6, the synchronization of multiple biaxial motion systems should be realized through continuous clock synchronization. In this thesis, we proposed an integrated continuous clock state and clock rate synchronization method to meet this requirement. This proposed method is discussed in this chapter.

Clock synchronization is a key issue in network based distributed multi-processor system, especially in applications that requires coordination and scheduling of events and message. As the clock of each node may drift over time, unsynchronized clock may arise. To overcome this problem, many clock synchronization methods have been proposed [41-49]. They can be broadly classified into two types, namely the external and internal clock synchronizations. External clock synchronization targets to synchronize the local clocks to an external, accurate global time. Its aim is to achieve better system *accuracy*, which is defined as the maximum deviation of local clock reading from real time [45]. Example of external clock synchronization includes Network Time Protocol (NTP) [46]. On the other hand, the internal clock synchronization targets to synchronize multiple local clocks, with no external time

source. Its aim is to get high system *precision*, which is defined as the maximum difference of simultaneous local clock readings.

For internal clock synchronization, one of the easiest strategies is the master-slave approach [42, 47]. In this approach, one or more nodes are assigned as the master nodes while the others as slave nodes. The master node periodically sends its clock state to the slave nodes for synchronization reference. This method requires minimum network traffic and processing power. However, it has a major problem of not being fault tolerant. Once the master is not performing well, the clock synchronization mechanism fails. Hence, it requires a very reliable master node.

Another common algorithm is the distributed approach with each node periodically reads the clock state of other nodes to calculate the reference clock state. As each node contributes towards the reference clock state, faulty nodes will cause wrong clock synchronization. To handle the faulty nodes, different fault-tolerant algorithms have been proposed. Examples include the convergence averaging algorithms [48-50], authentication protocols based algorithms [51, 52], atomic broadcast based algorithms [53, 54] and Byzantine agreement based algorithms [50] etc. These methods could prevent the faulty nodes from affecting the clock synchronization for limited number of faulty nodes.

Traditionally, clock synchronization algorithms synchronize only the clock state. Recently, methods that synchronize the clock rate have also been proposed [55-57]. For clock rate synchronization, it is often used together with the clock state synchronization to yield better clock precision.

As shown in Fig. 5-1, all the clock synchronization process involves two stages namely the acquisition stage and adjustment stage. Acquisition stage refers to the

process of obtaining the reference clock state and calculating the clock state adjustment term. The adjustment stage refers to the process of applying the adjustment to the local clock.

Fig. 5-1 Two stages of clock synchronization

The acquisition stage has been well studied [50-55]. In this thesis, we focus on the adjustment stage, which has strong implication on the continuity of the synchronized clock. Continuous clock synchronization is required in many applications, especially distributed real time applications. The key problem of a discontinuous clock is the missing or duplicating of time instances. This may cause undesired fault in applications, such as wrongly triggered events or measured events intervals.

There are two existing clock adjustment approaches, namely the instantaneous adjustment method and the amortization based adjustment method. In the instantaneous adjustment method, the clock state adjustment is applied instantaneously at the clock synchronization instance. This would generate a synchronized but discontinuous local clock. The amortization based adjustment method attempts to lessen this problem by spreading the clock state adjustment over a period of time. However, it can be shown that the method cannot eliminate the clock discontinuity. To overcome this limitation, a new method that performs clock state synchronization through clock rate instead of clock state adjustment is proposed. Furthermore, it is integrated with the clock rate synchronization to provide higher precision.

The outline of this chapter is organized as follows: In section 5.2, the basic of a clock model is presented. In section 5.3, the continuity performance of the two existing

clock state adjustment methods are reviewed and evaluated. To overcome the shortcoming of the existing methods, a new clock synchronization algorithm is proposed in section 5.4. Analysis on the precision of the algorithm is presented in section 5.5. The new algorithm has been implemented using microcontrollers. Its experimental results are presented in section 5.6. They have verified that the new approach performs better than the existing schemes. Section 5.7 concludes this chapter.

5.2 The Clock Model

Consider a typical digital clock signal generated by an oscillator that has a frequency f_o . This frequency corresponds to an oscillator resolution g , which is the time duration between two edges of the oscillator output signal:

$$g = \frac{1}{f_o} \tag{5.1}$$

Fig. 5-2 The generation of physical clock

In practice, the oscillator output is not used directly as the time source. Instead, a frequency divider and clock counter is used to obtain the desirable time base frequency from the oscillator output as shown in Fig. 5-2. The clock counter counts the number of oscillator pulses and generates an interrupt upon reaching a preprogrammed value. The counter is then reset to repeat the process. At each interrupt, a clock tick event is generated to yield the physical clock of the processor. The frequency divider and clock counter together provides an integer frequency division factor M . Thus, the physical clock's frequency f_c is obtained as

$$f_c = \frac{f_o}{M} \quad (5.2)$$

From (5.2), the physical clock resolution G can be obtained as

$$G = 1/f_c = M \times g \quad (5.3)$$

The physical clock $T_p(t)$ is the local unsynchronized clock. It is a function of the real time t and can be expressed as

$$T_p(t) = R(t) \times t \quad (5.4)$$

where $R(t)$ is the clock rate. Ideally, the clock rate should be unity. However, the oscillator output frequency f_o may drift from its nominal value $\overline{f_o}$. Consequently, the physical clock frequency $f_c(t)$ is different from its nominal value $\overline{f_c}$ and the clock rate becomes

$$R(t) = \frac{\overline{f_c}}{f_c(t)} \quad (5.5)$$

Substitute (5.2) into (5.5), we have

$$R(t) = \frac{M \times g}{\overline{G}} \quad (5.6)$$

where $\overline{G}$ represent the nominal or initial physical clock resolution. In (5.6), $\overline{f_c}$ is a constant for a given system, while $f_o(t)$ is generated by the oscillator and is not adjustable by software (similarly for g and $\overline{G}$). Consequently, the only adjustable parameter in (5.6) is M .

To describe the extent that the physical clock rate has drifted, a drift rate ρ is defined:

$$\rho(t) = 1 - R(t) \quad (5.7)$$

For a given system, a maximum drift rate $\rho_{\max}$ can be specified to determine whether a clock is functioning correctly. For a *good clock* [58], $T_p(t)$ must be a monotonically increasing function of real time with the clock rate bounded within the maximum drift rate:

$$|1 - R(t)| \leq \rho_{\max} \quad (5.8)$$

Besides the physical clock $T_p(t)$, a software maintained logical clock $T(t)$ can be constructed based on the physical clock and clock synchronization method. The logical clock is adjusted at each clock synchronization instance. We use $T^n(t)$ and $R^n(t)$ to denote the logical clock state and clock rate after the n^{th} clock synchronization.

Physical clock is maintained by the clock tick events. At each clock tick event, the physical clock is increased by G . Between the two adjacent clock ticks, the physical clock state remains unchanged. As the increment is the resolution G , the time between two clock ticks will never be used. Consequently, there is no time instance within G will be missed or repeated. Hence, the clock can be considered continuous.

For the logical clock, the clock may change with a size different from G due to software activities such as clock synchronization. If the increment is greater than G , some time instances may be missed. If the increment is negative, duplicated time instance would be generated. Such a clock is considered discontinuous clock. In summary, a clock is considered continuous if the clock state increments always equals to G . If there exists any increment not equals to G , the clock is considered discontinuous.

5.3 Evaluation of Existing Clock Adjustment Methods

As discussed in section 5.2, a clock state adjustment term ΔT_i for each node i can be determined from the acquisition stage of clock synchronization. In the adjustment stage, this term is updated into the local logical clock T_i . There are two updating approaches, namely the instantaneous approach and the amortization approach. In the following sections, these two methods are reviewed.

5.3.1 Instantaneous Clock Adjustment

In the instantaneous clock adjustment approach, once ΔT_i is obtained, the adjustment stage is immediately performed by adding ΔT_i to the logical clock. Define the clock state adjustment term ΔT_i^n as the difference between the reference clock $T_r^n(t_{0-}^n)$ and the local clock $T_i^n(t_{0-}^n)$, i.e.

$$\Delta T_i^n = T_r^n(t_{0-}^n) - T_i^n(t_{0-}^n) \quad (5.9)$$

where the superscript n represents the instant after the n^{th} synchronization, t_{0-}^n is the time just before the n^{th} synchronization instance. Then the new local logical clock after n^{th} synchronization can be obtained as

$$T_i^n(t) = T_i^{n-1}(t) + \Delta T_i^n \quad (5.10)$$

Since $T_i^0(t) = T_p(t)$, which is the unsynchronized physical clock of the local node, (5.10) can be rewritten as

$$T_i^n(t) = T_p(t) + \sum_{k=1}^n \Delta T_i^k \quad (5.11)$$

In (5.11), the clock state adjustment term ΔT_i^n can be larger than G and either positive or negative. Hence, the resulting clock is discontinuous. The clock state may leap forward or backward after clock synchronization causing missing or duplicating time instances.

5.3.2 Amortization Based Clock Adjustment

To overcome the discontinuity problem in the instantaneous clock adjustment approach, the amortization based clock adjustment approach has been developed [51, 52, 59, 60]. Instead of instantaneously change the logical clock, the adjustment term is spread over the accumulation interval A . In this case, the logical clock is adjusted at each clock tick event during the accumulation interval. The adjustment term is

$\frac{\Delta T_i^n}{A} \times G$, which eventually accumulates to ΔT_i^n within the interval A . The logical

clock at each clock tick after n^{th} synchronization is obtained as

$$\begin{cases} T_i^n(t_0^n) = T_i^{n-1}(t_0^n) \\ T_i^n(t_0^n + k \times G) = T_i^n(t_0^n + (k-1) \times G) + G + \frac{\Delta T_i^n}{A} \times G \end{cases} \quad (5.12)$$

where k is the number of clock ticks after the synchronization instance

$(k = 1, 2, 3, \dots, \frac{S}{G})$, S is the synchronization interval, t_0^n is the initial time after n^{th}

clock synchronization.

With amortization, the clock state increment is replaced by a series of smaller

adjustments equal to $(1 + \frac{\Delta T_i^n}{A})G$.

5.3.3 Limitations of the Methods

The amortization adjustment approach generates smoother synchronized clocks. However, it is not continuous. This is due to the nature of the “clock state adjustment” approach. As the clock state is adjusted by adding an adjustment term, either ΔT_i^n in the instantaneous method or $(1 + \frac{\Delta T_i^n}{A})G$ in the amortization method they result in clock state increments which are different from the clock resolution G . Hence, both methods produce discontinuous clocks, as defined in section 5.2.

Fig. 5-3 Illustration of discrete clock synchronization and amortization

An example of clock synchronization with the two clock state adjustment methods is shown in Fig. 5-3. In this example, an adjustment term ΔT_i^n is obtained for the n^{th} synchronization. The instantaneous and amortization based clock adjustment methods are applied assuming the clock is initially synchronized after the $(n-1)^{th}$ synchronization. From Fig. 5-3, it is shown that the synchronized clock by the instantaneous adjustment method causes large jump discontinuity at each

synchronization instance, while the synchronized clock by the amortization based clock adjustment method consists a number of small jump discontinuities after the clock synchronization instance. As both methods result in discontinuous clocks, an approach that can generate continuous clock is desirable. This is presented in the next section.

5.4 The Proposed Continuous Integrated Clock Synchronization

In this study, a new method that produces continuous clock synchronization using clock rate adjustment is proposed. The method first uses the clock rate adjustment to synchronize the clock state to yield a continuous local logical clock. It is then integrated with the clock rate synchronization to further improve the performance.

5.4.1 Continuous Clock State Synchronization by Clock Rate Adjustment

Similar to other methods, we utilize the clock adjustment term ΔT_i^n obtained from the clock acquisition stage to adjust the logical clock. However, instead of being directly changed, the clock state is adjusted indirectly through clock rate adjustment. Since there is no direct change to the clock state, it will not introduce extra clock jump, causing discontinuous clock.

The key idea of this method is to adjust the clock rate such that the accumulated change of the clock state due to clock rate adjustment equals to the adjustment term ΔT_i^n over a time interval A . The desired adjustment applied to the clock rate is then defined as clock state adjustment rate Δr_i^n , which is calculated as

$$\Delta r_i^n = \frac{\Delta T_i^n}{A} \quad (5.13)$$

At each synchronization instance, the clock rate $R_i^n(t)$ is adjusted as:

$$R_i^n(t) = R_i^{n-1}(t) + \Delta r_i^n = R_i^{n-1}(t) + \frac{\Delta T_i^n}{A} \quad (5.14)$$

Since the adjustment term ΔT_i^n is applied indirectly through clock rate adjustment, it will not introduce extra clock jump. Thus, a continuous clock as defined in section 5.2 can be achieved.

5.4.2 Clock Rate Synchronization

Clock rate synchronization is used to synchronize the clock rates of different clocks to reduce drift. It is performed by adjusting the clock rate with a rate adjustment term ΔR_i^n . This is calculated based on a reference clock rate $R_r^n(t)$ and the local clock rate $R_i^n(t)$, i.e.

$$R_i^n(t) = R_i^{n-1}(t) + \Delta R_i^n \quad (5.15)$$

where ΔR_i^n is the clock rate adjustment term at the n^{th} clock rate synchronization

$$\Delta R_i^n = R_r^n(t_{0-}^n) - R_i^n(t_{0-}^n) \quad (5.16)$$

The difference of the clock rate synchronization as compared to the clock state synchronization is mainly on obtaining the clock rate. Unlike the clock state, that is directly accessible, the clock rate is not explicitly measured. The details of the clock rate representation and adjustment will be covered in section 5.6.

5.4.3 Integrated Continuous Clock State and Clock Rate Synchronization

The proposed continuous clock state synchronization method aims to produce continuous clock, while clock rate synchronization reduces the drift rate. Integrating the two methods would give a better overall performance. Since both methods utilize clock rate adjustment, it is more efficient to combine them together. For simplicity, the

clock rate synchronization interval S_r and the clock state synchronization interval S_c are set to be the same, i.e. $S_c = S_r = S$. Thus, from (5.14) and (5.15),

$$\begin{cases} R_i^n(t) = R_i^{n-1}(t) + \frac{\Delta T_i^n}{A} + \Delta R_i^n, & t_0^n \leq t < t_0^n + A \\ R_i^n(t) = R_i^{n-1}(t) + \Delta R_i^n, & t_0^n + A \leq t < t_0^n + S \end{cases} \quad (5.17)$$

There are two ways to obtain the reference clock rate $R_r^n(t)$. One way is to implement a separate reference clock rate acquisition stage similar to the reference clock state acquisition. It is straight forward but needs extra network traffic and processing power. Another method is to utilize the clock state adjustment term to estimate the reference clock rate. It requires no extra network traffic and is used in this thesis.

The local clock rate can be estimated from the change of clock state as

$$R_i^n(t) = \frac{T_i^n(t_0^n) - T_i^{n-1}(t_0^n)}{S} \quad (5.18)$$

Similarly, the reference clock rate is obtained as

$$R_r^n(t) = \frac{T_r^n(t_0^n) - T_r^{n-1}(t_0^n)}{S} \quad (5.19)$$

Substitute (5.18) and (5.19) into (5.16), the clock rate adjustment is obtained as

$$\begin{aligned} \Delta R_i^n &= \frac{T_r^n(t_0^n) - T_r^{n-1}(t_0^n)}{S} - \frac{T_i^n(t_0^n) - T_i^{n-1}(t_0^n)}{S} \\ &= \frac{T_r^n(t_0^n) - T_i^n(t_0^n)}{S} - \frac{T_r^{n-1}(t_0^n) - T_i^{n-1}(t_0^n)}{S} \\ &= \frac{\Delta T_i^n - \Delta T_i^{n-1}}{S} \end{aligned} \quad (5.20)$$

Substitute (5.20) into (5.17), we have

$$\begin{cases} R_i^n(t) = R_i^{n-1}(t) + \frac{\Delta T_i^n}{A} + \frac{\Delta T_i^n - \Delta T_i^{n-1}}{S}, & t_0^n \leq t < t_0^n + A \\ R_i^n(t) = R_i^{n-1}(t) + \frac{\Delta T_i^n - \Delta T_i^{n-1}}{S}, & t_0^n + A \leq t < t_0^n + S \end{cases} \quad (5.21)$$

If the synchronization interval is set to be the same as accumulation interval, i.e. $A=S$, then (5.13) can be represented as

$$\Delta r_i^n = \frac{\Delta T_i^n}{A} = \frac{\Delta T_i^n}{S} \quad (5.22)$$

Substitute (5.22) into (5.21), we have

$$\begin{aligned} R_i^n(t) &= R_i^{n-1}(t) + \frac{\Delta T_i^n}{S} + \Delta R_i^n \\ &= R_i^{n-1}(t) + \frac{\Delta T_i^n}{S} + \frac{\Delta T_i^n - \Delta T_i^{n-1}}{S} \\ &= R_i^{n-1}(t) + \frac{2\Delta T_i^n - \Delta T_i^{n-1}}{S} \end{aligned} \quad (5.23)$$

In (5.23), the clock state adjustment rate and clock rate adjustment term are integrated.

We use $\tilde{R}_i^n$ to represent the integrated clock rate adjustment term

$$\tilde{R}_i^n = \frac{2\Delta T_i^n - \Delta T_i^{n-1}}{S} \quad (5.24)$$

In the next section, analysis and experiment will be performed in the next section to verify the performance.

5.5 Precision Analysis

5.5.1 Clock State Synchronization with Instantaneous Clock Adjustment

In the acquisition stage of the clock synchronization, the clock state adjustment term is obtained by taking the difference of the reference clock state $T_r^n(t_{0-}^n)$ and the local clock $T_i^n(t_{0-}^n)$. However, the receiving and calculation of $T_r^n(t_{0-}^n)$ would result in delay

due to transmission propagation, data reading and computation. We define E as the total delay, then the actual reference clock state $T_{ra}^n(t_{0-}^n)$ is defined as:

$$T_{ra}^n(t_{0-}^n) = T_r^n(t_{0-}^n) + E^n \quad (5.25)$$

where E^n is the delay corresponds to reference clock state $T_r^n(t_{0-}^n)$ for the n^{th} synchronization.

To reduce the effect of the delay E , an estimated constant delay $\tilde{E}$ is added to the reference clock state for compensation. Hence, the estimated reference clock state $T_{re}^n(t_{0-}^n)$ is defined as

$$T_{re}^n(t_{0-}^n) = T_r^n(t_{0-}^n) + \tilde{E} \quad (5.26)$$

$T_{re}^n(t_{0-}^n)$ is then used in (5.9) to replace $T_r^n(t_{0-}^n)$ in calculating the clock state adjustment term. The difference between the estimated and actual delay is denoted as ε^n and

$$\varepsilon^n = T_{ra}^n(t_{0-}^n) - T_{re}^n(t_{0-}^n) = E^n - \tilde{E} \quad (5.27)$$

From (5.27), the delay estimation error ε^n depends on the variance of the total delay E . It cannot be eliminated by clock synchronization. We denote $\delta^n(t)$ as the clock state error of the logical clock with respect to the reference clock in the n^{th} synchronization interval. Within each synchronization interval, the initial value of the clock state error $\delta^n(t_{0+}^n)$ equals to ε^n in a conventional clock state synchronization method, where t_{0+}^n is the time instance immediately after the n^{th} clock synchronization instance. The clock state error $\delta^n(t)$ continues to increase from ε^n in the interval due to clock drift. Hence, the expression of $\delta^n(t)$ over time t after the n^{th} synchronization is defined as

$$\delta^n(t) = \rho^n \times (t - t_0^n) + \varepsilon^n \quad (5.28)$$

where ρ^n is the clock rate after the n^{th} synchronization, $\delta^n(t)$ represents the clock state error after the n^{th} synchronization

The maximum clock state error occurs right before the next clock synchronization instance. At this point of time, the clock state errors include the initial error ε^n , as well as the accumulated error due to drift ρ in the synchronization interval S , i.e.

$$\delta_{\max}^n = \rho^n \times S + \varepsilon^n \quad (5.29)$$

For the overall system, the maximum clock state error is derived from the worst case scenario of (5.29)

$$\delta_{\max} = \rho_{\max} \times S + \varepsilon_{\max} \quad (5.30)$$

Since $\delta_{\max}$ could be positive or negative, the precision π of multi-node synchronized system is twice the individual maximum clock state error.

$$\pi = 2\delta_{\max} = 2\rho_{\max} \times S + 2\varepsilon_{\max} \quad (5.31)$$

5.5.2 Integrated Clock State and Clock Rate Synchronization

By integrating clock rate synchronization into clock state synchronization, the precision is changed since the drift rate ρ is changed due to clock rate synchronization. In clock rate synchronization, the reference clock rate is calculated based on the reference clock as in (5.19). The actual reference clock rate $R_{ra}^n(t)$ is defined as

$$R_{ra}^n(t) = \frac{T_{ra}^n(t_0^n) - T_{ra}^{n-1}(t_0^{n-1})}{S} \quad (5.32)$$

Since the estimated reference clock state $T_{re}^n(t_{0-}^n)$ is used instead of the actual reference clock, the estimated clock rate $R_{re}^n(t)$ is

$$R_{re}^n(t) = \frac{T_{re}^n(t_{0-}^n) - T_{re}^{n-1}(t_{0-}^{n-1})}{S} \quad (5.33)$$

After clock rate synchronization based on the estimated reference clock rate, there still exist a clock drift rate ρ^n due to the estimation error. Assuming the clock rate is maintained constant within the synchronization interval after the clock rate synchronization, ρ^n can then be derived from (5.27), (5.32) and (5.33) as

$$\begin{aligned} \rho^n &= R_{ra}^n(t) - R_{re}^n(t) \\ &= \frac{T_{ra}^n(t_{0-}^n) - T_{ra}^{n-1}(t_{0-}^{n-1})}{S} - \frac{T_{re}^n(t_{0-}^n) - T_{re}^{n-1}(t_{0-}^{n-1})}{S} \\ &= \frac{T_{ra}^n(t_{0-}^n) - T_{re}^n(t_{0-}^n)}{S} - \frac{T_{ra}^{n-1}(t_{0-}^{n-1}) - T_{re}^{n-1}(t_{0-}^{n-1})}{S} \\ &= \frac{\varepsilon^n - \varepsilon^{n-1}}{S} \end{aligned} \quad (5.34)$$

Substitute (5.34) into (5.29), the clock state error is obtained as

$$\delta_{\max}^n = \frac{\varepsilon^n - \varepsilon^{n-1}}{S} S + \varepsilon^n = 2\varepsilon^n - \varepsilon^{n-1} \quad (5.35)$$

In the worst case, $\varepsilon^n = \varepsilon_{\max}$ and $\varepsilon^{n-1} = -\varepsilon_{\max}$, we have

$$\delta_{\max} = 3\varepsilon_{\max} \quad (5.36)$$

Substitute (5.36) into (5.31), the precision of the proposed system is

$$\pi = 6\varepsilon_{\max} \quad (5.37)$$

Comparing (5.31) and (5.37), it is observed that the proposed method eliminates the $\rho_{\max} \times S$ term that exists in the clock state synchronization method. Consequently, the precision of the proposed system is independent on the synchronization interval. Thus,

the synchronization interval can be chosen to be larger without affecting the precision. This has the advantage of reducing the network traffic and processing power needed for clock synchronization. Oscillators with larger drift rate $\rho_{\max}$ are also acceptable as it does not affect the precision.

The analysis assumes that the oscillator drift rate is approximately constant within the synchronization interval. Hence, the precision bound is invalid if the synchronization interval chosen is too long that there is significant variation in the drift rate within that interval.

5.6 Implementation and Experimental results

To implement the proposed method, we use (5.23) to adjust the local clock rate. From (5.6), the clock rate can be adjusted by changing the frequency division factor M since the other two parameters g and $\bar{G}$ are not adjustable. Substituting (5.6) into (5.23), the clock rate adjustment equation becomes

$$\frac{M_i^n \times g}{\bar{G}} = \frac{M_i^{n-1} \times g}{\bar{G}} + \frac{2\Delta T_i^n - \Delta T_i^{n-1}}{S} \quad (5.38)$$

where M_i^n represents the frequency division factor of node i after the n^{th} synchronization

From (5.38), we have

$$M_i^n = M_i^{n-1} + \frac{2\Delta T_i^n - \Delta T_i^{n-1}}{S} \times \bar{M}_i \quad (5.39)$$

where $\bar{M}_i = \bar{G}/g$ is the nominal value of M_i , as well as the initial unsynchronized value. The equation (5.39) is effectively equivalent to (5.23). Both represent the clock rate adjustment for the proposed clock synchronization scheme.

For ease of implementation, we let

$$\Delta M_i^n = \frac{2\Delta T_i^n - \Delta T_i^{n-1}}{S} \times \bar{M}_i \quad (5.40)$$

Hence, (5.39) can be rewritten as

$$M_i^n = \bar{M}_i + \sum_{k=1}^n \frac{2\Delta T_i^k - \Delta T_i^{k-1}}{S} \times \bar{M}_i = \bar{M}_i + \sum_{k=1}^n \Delta M_i^k \quad (5.41)$$

From (5.41), the proposed adjustment is implemented by adding the term $\sum_{k=0}^n \Delta M_i^k$ into the initial frequency division factor $\bar{M}_i$.

Since M_i is an integer, only $\bar{M}_i$ number of discrete levels of clock rate can be used for clock adjustment. This limits the performance of the clock rate synchronization. To increase the flexibility of M , [55, 61] introduced a hardware based none-integer frequency divider method that produces non-integer frequency division factor M . In this thesis, we propose a different approach without additional hardware. In (5.41),

$\sum_{k=1}^n \Delta M_i^k$ is a real number. Since the frequency division factor is an integer, the fraction part of $\sum_{k=1}^n \Delta M_i^k$, denoted as M_f , will be neglected when being added to $\bar{M}_i$.

This will lower the synchronization precision. In this study, it is proposed that the fraction part be accumulated over an interval I , e.g. 1000G instead of being discarded. This can be done easily by multiplying the fraction with the interval length. The accumulated value $\sum M_f$ can be rounded upto an integer. This integer adjustment term is then divided into $\sum M_f$ number of ones and evenly distributed into the interval I .

To evaluate the performance of the proposed approach, an experiment has been performed. A three-node distributed embedded system connected with a CAN bus has been setup. Master-slave based internal clock synchronization is adopted for the acquisition stage. One node is assigned as the master with its clock being used as the reference clock. By utilizing the broadcasting property of CAN, as well as its priority contention based collision arbitration method, the reference clock is delivered to other nodes with a deterministic time delay [34]. If the synchronization frame is broadcasted with high priority, the propagation delay will be small and results in a very small delay estimation error ε . The estimated propagation delay $\tilde{E}$ is obtained from the first broadcasting after initialization.

For the experiment, the clock synchronization interval, clock rate synchronization interval and accumulation interval are chosen to be the same. Hence, the adjustment equation (41) can be used for the proposed integrated clock synchronization. In the experiment, the oscillator has a nominal frequency of 10MHz, corresponding to a nominal oscillator resolution $\bar{g}=0.1\mu\text{s}$. The initial value of the frequency division factor $\bar{M}$ is 500, resulting in a nominal physical clock resolution $\bar{G}=50\mu\text{s}$.

Fig. 5-4 shows the experimental results of the proposed method and the clock state synchronization method with instantaneous clock adjustment. The proposed method uses (5.23) for clock adjustment while the conventional clock state synchronization follows (5.10). To have fair comparison, both methods use the same synchronization interval i.e. 250 sec in the experiment. From the figure, it is clear that the proposed method gives a continuous logical clock. Furthermore, it also greatly increases the clock precision. This result agrees well with the analysis presented in section 5.5.

Fig. 5-4 Clock errors of the proposed method and conventional method

To evaluate the effect of synchronization interval on the clock precision for the proposed approach, three set of experiments with synchronization intervals of 125sec, 250sec and 500sec have been performed. The experimental results are shown in Fig. 5-5. It is observed that steady state arrives after the first two synchronization instance. From the results, the maximum steady state clock state errors are determined and listed in Table 5-1. It is observed that there is no significant difference among the three experiments. This verifies the analysis that the precision achieved from the proposed clock synchronization is independent on the synchronization interval. Fig. 5-6 shows the similar experiments for the conventional clock state synchronization method. The clock state error shows clearly the dependency on the synchronization interval, which follows the analysis from (5.31).

Fig. 5-5 Clock errors at different synchronization intervals (proposed method)

Fig. 5-6 Clock errors at different synchronization intervals (conventional method)

Synchronization interval	S=125 Sec	S=250 Sec	S=500 Sec
Maximum steady state clock error	107 us	106 us	92 us

Table 5-1 Steady state clock errors at different synchronization interval

5.7 Conclusions

An integrated clock synchronization algorithm that combines the continuous clock state synchronization by clock rate adjustment and the clock rate synchronization is proposed in this chapter. The proposed method is capable of producing continuous synchronized local clock. It overcomes the shortcomings of the existing approaches such as the instantaneous adjustment and amortization based clock synchronization. Furthermore, the proposed method has a higher precision. Analysis has shown that the precision of the proposed method is independent of the clock synchronization interval. This characteristic allows us to select a larger interval that results in fewer networks traffic and processing power while achieving the same precision. Experimental results have validated the analysis. The proposed approach can easily be realized using embedded systems that support access to the oscillator frequency divider and counter. By implementing the proposed clock synchronization method into the network based motion control system, motion synchronization among different nodes can be achieved.

Chapter 6

Prototyping and Implementation

6.1 Overview

In the previous chapters, the approaches for extending the flexibility and reliability of the network based motion control system have been presented. New motion control algorithms and network protocols have been proposed. A prototype has been constructed with the proposed approaches. Various experiments have been performed to evaluate the system. In this chapter, details of the prototype are presented and various implementation issues are discussed.

The constructed prototype is a scaled down version of the intended application. It consists of two biaxial mechanisms, two local motion controllers, a central controller and a CAN bus. The block diagram of the prototype is shown in Fig. 6-1. A picture of the prototype is shown in Fig. 6-2. As shown in the figures, the two biaxial mechanical mechanisms are controlled by the two local motion controllers respectively. These two controllers are connected to the central controller through a CAN bus. The commands and desired motion trajectories are delivered from the central controller to the local motion controllers through the network.

Fig. 6-1 Block diagram of the prototype

Fig. 6-2 A picture of the prototype

In the following sections, the major parts of the prototype are presented in detail. In section 6.2, the mechanical design of the biaxial motion system is discussed. In section 6.3, the hardware and firmware of the local motion controller are presented. The central controller is presented in section 6.4. The network system with a large number of nodes is discussed in 6.5. Experimental results of the prototype with water spray are presented in 6.6. Section 6.7 concludes this chapter.

6.2 Design of the Biaxial Motion Mechanism

A biaxial motion mechanism is built with two brushless servo motors and the corresponding driving amplifiers. This biaxial motion mechanism follows the structure described in chapter 3. As shown in Fig. 6-3, two servo motors are placed perpendicular to each other. The rotating motor is mounted on the axis of the base motor. The axis of the rotating motor is defined as the ϕ axis, while the axis of the base motor is defined as the θ axis. A nozzle is placed on the ϕ axis and becomes the end-effector of the mechanism.

Fig. 6-3 The biaxial motion mechanism

Both of the two motors have limited rotation ranges from 0 to π rad. Hence, the motion space of the end-effector is on the upper half spherical surface. This range limitation is imposed as the nozzle only operates above the water level while the mechanism is submersed below the water level. The setup coupled with the range

limitation also avoids cable entanglement. The power cable, feedback cable and water hoses creates physical constraint to the axes.

The servo motors used in the prototype are Fuji FALDIC-W series GYS101DC2-T2A 100W brushless motor [62]. Each of the motors is equipped with a 32768 pulse per revolution (PPR) incremental encoder, capable of providing a maximum of 17 bit resolution position feedback. A servo amplifier is used to drive the motor. The amplifier provides power amplification, speed control loop and current control loop for the servo motor. Therefore, only position control loop is required on the local motion controller.

6.3 Local Motion Controller

The local motion controller consists of a controller board and the corresponding firmware. In the following sections, the hardware and firmware design are presented.

6.3.1 Hardware Design

The hardware of the local motion controller is a controller board. The core of this controller board is a Microchip PIC18 microcontroller, which is the main processor of the board. Other functional blocks include encoder interface, amplifier interface, CAN driver, voltage regulation unit, pushbuttons, status indicators etc. Fig. 6-4 shows an overview of the functional blocks of the controller board.

Fig. 6-4 Functional blocks of the controller board

The prototype of the controller board is shown in Fig. 6-5. The prototype is constructed by utilizing a Microchip PIC18F6680 demo board and a wire-warped expansion interface board. The schematic diagram of the controller board is given in Appendix B.

Fig. 6-5 The controller board prototype

To meet the requirement of the application, the Microchip PIC18F6680 microcontroller [63] is chosen based on its processing capability and peripherals. The PIC18F6680 is an 8 bit, RISC microcontroller. It can support upto 40 MHz clock

input, equivalent to 10M MIPS operation speed [63]. It also has several useful integrated peripheral features such as 4 timers, 2 PWM output, CAN controller, SPI interface, I2C interface and USART module.

The amplifier interface includes both digital and analog I/Os between the controller board and the servo amplifiers. Digital outputs to the amplifiers are used for controlling the operating state of the amplifiers, e.g. on, off, standby, while the analog outputs are the control signals for the amplifiers. As shown in Fig. 6-6, a dual channel 12-bit Digital Analogue Converter (DAC) module is used to provide analog outputs to the amplifiers. The analog signal ranges from -10 to 10 volt, which corresponds to the motor speed range from -300 rpm to 300 rpm. A level translator is used to convert the different digital voltage levels between the controller board and amplifiers. The schematics diagram of the amplifier interface circuit is given in Appendix B2.

Fig. 6-6 Block diagram of the amplifier interface

The encoder interface receives the quadrature signal from the servo motor's incremental sensor and converts it to position count. As shown in Fig. 6-7, the differential quadrature signal is first converted to TTL type. It is then converted to position counts through quadrature decoder. The position count is then sent to the microcontroller through a parallel port. The HTCL2032 quadrature decoder and

MC3468 differential line receiver have been used for the interfacing circuit. The schematics diagram of the encoder interface circuit is given in Appendix B1.

Fig. 6-7 Block diagram of the encoder interface

An integrated CAN controller peripheral is included in the PIC18F6680 microcontroller. By utilizing this CAN controller and adding a CAN transceiver, a CAN node can be configured. The CAN node can receive and transmit CAN messages with both CAN1.0 and CAN2.0 standards. The CAN controller provides comprehensive features including masks, filters, receive and transmit buffers. The details of the CAN features can be found in [63].

Other peripherals including voltage regulation unit, status indicators and in-circuit serial programming (ICSP) port are also implemented on the controller board. The voltage regulation unit provides regulated DC power supply to the whole board. The pushbuttons are designed for local manual control. The ICSP is the port used for microcontroller programming and debugging.

6.3.2 Firmware Design and Implementation

The controller firmware implements the functions such as closed loop motion control, CAN message receiving and transmitting, command execution, and data handling. It is developed in C language with Microchip C18 compiler.

The program contains a main routine and two Interrupt Service Routines (ISRs). The main routine is an infinite loop, which includes the message transmission, command execution and data handling functions. The two ISRs are the motion control ISR and

the CAN message receiving ISR. In this project, a timer interrupt is used for the motion control ISR. While a CAN receive interrupt is used for the CAN message receiving ISR. Two interrupt priorities are available in the PIC18F6680 microcontroller. For this study, the motion control has tighter timing constraint. Hence, a higher interrupt priority is assigned to the motion control ISR. In the following sections, the main routine and the two ISRs are discussed in detail.

6.3.2.1 Main Routine

The main routine includes the less time-critical functions of the program including command execution, motion data handling, trajectory up-sampling and message transmission. The routine consists of a program initialization process and an infinite loop operation. A flowchart of the main routine is shown in Fig. 6-8.

Fig. 6-8 Flow chart of the main routine

As shown in the figure, execution of the functions in the main routine depends directly or indirectly on the states of the ISRs. Status flags are used to control the execution of the functions in the infinite loop. In the loop, status flags determine whether the corresponding function executions are needed. The status flags are accessible and set by the two ISRs. Based on their operating status, the two ISRs may set the flags to activate the corresponding functions. The flags are cleared after completion of the function execution.

Once a new message has been received by the CAN message receiving ISR, the corresponding flags are set by the ISR. The main routine will then decode the received message and decide on the types of the received data. As shown in Fig. 6-8, if the received data is a command, it is executed immediately. If the received data is motion data, data handling function is executed. The data handling function implements the receiver part of the motion data streaming protocol. In this project, a buffer of 30 position samples is used. The data handling function manages the buffer by storing validated motion data into the buffer. Each received motion data need to pass the sequence control before being stored into the buffer. Those do not pass are discarded and a retransmission request is sent to the central controller. Implementation of the sequence control follows the protocols presented in Chapter 4.5.

When a message transmission is requested, the message transmission function is executed. The function is responsible for encoding the application data into CAN message. The implementation follows the protocol specified in Chapter 4.4.

Desired trajectory up-sampling by interpolation is also implemented in the main routine. In the system, the received motion data can not be used directly for control as they are the down-sampled data. Hence, up-sampling is performed to reconstruct the

original desired trajectory. Up-sampling is not performed in the motion control ISR, since it involves significant computation time affecting the ISR performance. The interpolated motion data are stored in a stack. Motion data that has already been used for motion control is removed from the stack. Whenever the number of motion data left in the stack is less than 5 samples, a new up-sampled is performed to generate 5 more up-sampled motion data. The up-sampling follows the protocol presented in Chapter 4.5.5.

6.3.2.2 Motion Control ISR

The proposed cross-coupled controller has been implemented as an ISR with the period set as 5ms. A flow chart of the motion control ISR is shown in Fig. 6-9.

Fig. 6-9 Flow char of the motion control ISR

As shown in the figure, the present and desired positions are first obtained in the routine. Then the PID or CCC control signal calculation function is executed based on the control mode selection. The calculated control signal is then sent to the amplifier through the amplifier interface. This signal becomes the speed reference for the amplifier.

The calculation of control signal in the PID is shown in Fig. 6-10. Three controller gains, namely the proportional gain, integral gain and derivative gain are predefined based on the controller design. All the calculations are performed in fixed point. The computation time takes only 0.2ms.

Fig. 6-10 Flow chart of the PID calculation function

The CCC routine is similar to the PID, except that the calculation is performed in the planar trajectory space. Fig. 6-11 shows the flow chart of the CCC calculation function. As shown in the flow chart, the transformation from the spherical space into the planar trajectory space is performed first. The calculated control signal is then

transformed back to the spherical space. Calculation of the control signal follows the same way as the PID control, since the CCC controller used in the planar trajectory space is also a PID controller.

Fig. 6-11 Flow chart of the CCC calculation function

6.3.2.3 CAN Message Receiving ISR

An ISR is implemented to receive CAN messages. It is also responsible to detect communication errors. As introduced in section 6.3.1, the CAN controller has 6 receive message buffers. Once a CAN message is received from the bus, the CAN controller stores the message into one of the receive buffers and interrupts the processor. The ISR will then retrieve the CAN message from the buffer. A flag is then set to inform the main routine that a new data has been received. Since the CAN controller has 6 buffers, it is tolerable for a delay of executing the ISR after a message

has been received on the CAN controller. Therefore, the interrupt for the CAN message receiving ISR is set at a lower priority.

6.4 Central Controller Design

The system uses a central controller to control the local motion nodes through the CAN bus. This central controller includes a computer, a CAN interface card and the controller software. The CAN interface card used in this project is the PEAK PCAN-PCI interface card [64]. It has dual CAN channels and a PCI interface. The controller software is written to implement the central control algorithm and network. It also provides a Graphical User Interface (GUI) for controlling and monitoring the system. The trajectory generation function is also implemented on this software. In the following sections, the software details will be introduced.

6.4.1 Central Controller Software Overview

An overview of the central controller software is illustrated in Fig. 6-12. As shown in the figure, the software includes four major functional blocks: the central controller, the graphical user interface, the trajectory generation and the communication interface. Data storage is also used for storing the desired trajectories as well as feedback data.

As shown in Fig. 6-12, the GUI is the interface for user control and monitoring. Command generated from the GUI is delivered to the controller. The trajectory generation function generates the desired trajectories based on the trajectory specifications, which is specified by the user through the GUI. Based on the user commands and desired trajectory, the controller sends out the commands and motion data to the local nodes. The communication interface converts these data into CAN

messages and transmits them to the nodes. In the following sections, each of these four blocks is introduced.

Fig. 6-12 Functional blocks of the central controller

6.4.2 Graphical User Interface

The GUI includes user interactive fields for controlling and monitoring the system. Fig. 6-13 shows a snapshot of the GUI and its various fields. There are control pushbuttons, options and parameters fields for controlling purpose, as well as status display field for monitoring purpose. Furthermore, the desired trajectory specification field for trajectory generation is also included.

Fig. 6-13 The Graphical User Interface (GUI)

The control pushbutton field is used for accepting user commands, e.g. to start, stop, or pause a desired motion. The option field provides the users a number of options on the system, such as the control mode, controller method etc.

The parameter field is used for updating the operating parameters for the local motion nodes. These parameters include controller gains of the PID and CCC controllers, starting point and adjustment of the encoders etc. By activating the “Update Parameter” button after filling in the parameters, the new parameters will be sent to the desired nodes to replace the old parameters.

In the trajectory specification field, various desired trajectory specifications can be entered. The desired trajectory can then be generated through the trajectory generation block. More details are given in section 6.4.5. The GUI also has a status field for monitoring the system. Such as the network status, motor status, error statistics etc.

6.4.3 Controller Block

The controller block is responsible for managing and controlling the system based on the user command entered from the GUI. It translates the user command and the selected desired trajectory into messages that the local motion nodes can understand. These messages include the command messages and periodic motion data messages. The controller also receives feedbacks from the local nodes and reacts accordingly.

The motion data down-sampling (4.5.1) and motion scheduler (4.5.2) of the motion data streaming protocol are implemented in this block. The controller down-samples the desired trajectory and assigns a sequence number to each motion data. These motion data are then sent to the local nodes in sequence. If requests are received from the local nodes, the controller then responds accordingly based on the Table 4-2. Doubled transmission rate is achieved by sending two motion data in one transmission period. Halved transmission rate is achieved by sending only one data over two transmission periods

Since motion data are required to be transmitted to each motion node periodically, the software needs to generate periodic events. The period depends on the sampling period and the down-sampling factor. As mentioned in Chapter 2, the sampling rate is 5ms and the down-sampling factor is 5. Hence the transmission period is 25ms. In this implementation, a multimedia timer [65] is used to provide the periodic events at the frequency of 40Hz, as the system timer has poor precision in the mini-second level [65]. Multimedia timer can provide a much higher precision with a 1ms resolution.

Since the controller is periodic and time dependent, the performance is sensitive to timing accuracy. If the controller is implemented in the same thread of other parts of the program e.g. GUI, it may be delayed by their activities. To overcome this problem,

the controller is implemented as a different thread to other parts of the software so that they can exist in parallel and do not affect each others.

6.4.4 Communication Interface

The communication interface is responsible for transmitting and receiving CAN messages. It provides the interface to the CAN hardware and retrieves the CAN messages. It also encodes and decodes the CAN messages for transmitting and receiving respectively.

The CAN-PC interface card has a Dynamic Link Library (DLL) for device Application Programming Interface (API). It is implemented in the communication interface. By using the API, the central controller can send and receive CAN messages through the CAN-PC interface card.

The message encoding, decoding and scheduling protocol are also implemented in this block. In the transmitting mode, the values of each CAN message field are assigned. The data field is assigned based on the application data, following the format defined in Chapter 4.4.1. The identifier field is then assigned based on the message priority, deadline and receiver ID, following the rules in Chapter 4.4.2. For example, the retransmitted motion data and normal motion data are of the same type. Hence, they have the same priority. But retransmitted data have a tighter deadline than the normal motion data. Consequently, the retransmitted data has a higher priority level.

6.4.5 Trajectory Generation

The trajectory generation function is also implemented on the central controller. It generates the desired trajectories for each axis of the biaxial motion system based on the trajectory transformation method presented in Chapter 3.4. The original trajectories

are entered by the user from the GUI. Options for filtering the original and transformed trajectories are available. The trajectory generation is performed offline and stored as text files.

The trajectory generation is simple to use. The user needs not know the actual equation. Instead, the user simply fills in the fields on the GUI as shown in Fig. 6-14. The field on the left hand side specifies the original trajectory. Users need to fill in the time and corresponding θ and ϕ axis angles of the trajectory. Sampling times and filtering options are available to the users. The user then activates the “Generate Transformed Map” button. The desired position and speed trajectories are generated and automatically saved in files.

Fig. 6-14 The trajectory generation field

6.5 Network System with Large Numbers of Nodes

The experiments on the network system reported in the previous discussions are all conducted with the two bi-axial fountain heads the prototype. However, the number of fountain heads connected in the actual application may be upto to 100. On the other hand, it is impractical to setup a full 100-node prototype for experiment. To verify that the proposed network system works well with 100 fountain heads, the throughput and utilization rate of the network are investigated and analyzed in this section.

In the following discussions, the throughput is defined as the amount of data delivered over a network/data-link per time unit. It is usually measured in bit per second (bps). The throughput is always smaller than the data-link bit rate (channel capacity) due to the overheads. By comparing the throughput with the bit rate of the network/data-link, the utilization rate can be obtained as follows:

$$\text{Utilization rate} = \frac{\text{Throughput}}{\text{Bit rate}}$$

In a point-to-point or point-to-multipoint communication system, the utilization rate can be very near to 100%. For a multiple point communication such as the CAN bus, the maximum utilization rate may be significantly smaller than 100% due to the flow control, packet loss etc.

In this section, we analyze the throughput and utilization rate of the CAN bus for our system. As shown in Table 6-1, the throughputs of the 2-fountain-head prototype and the intended 100-fountain-head system for fountain application are calculated based on the data transmission of all nodes on the CAN bus.

Number of local nodes	Sent from central controller				Sent from the local controllers				Total throughput (kbps)	Utilization rate at 500kbps bit rate (%)	Utilization rate at 1Mbps bit rate (%)
	Message size m_c (bit)	Data frequency f_c (Hz)	Number of messages per transmission n_c	Throughput (kbps)	Message size m_l (bit)	Data frequency f_l (Hz)	Number of messages per transmission n_l	Throughput (kbps)			
2	84	40	2	6.72	44	0.1	1	0.009	6.73	1.35	0.67
100	84	40	100	336	44	0.1	1	0.44	336.4	67.3	33.6

Table 6-1 comparison of throughputs for 2 and 100 fountain heads system

As introduced in Section 4.2.2, the data are transmitted in messages using the CAN communication. Each message includes a header and upto 8 bytes of in data field. In the proposed network system, the message data fields consist 8 bytes and 3 bytes respectively for the central controller and local controller. Since the standard CAN

header is 20 bits, their corresponding messages sizes m_c and m_l are 84 bits and 44 bits respectively.

Since most of the data transmitted by the central controller are motion data, they are all periodic. According to (4.2), the transmission frequency equals to the sampling frequency f_s divided by the down-sampling factor n , which is equal to 5 as discussed in Section 4.5.1. Hence

$$f_c = \frac{f_s}{n} = \frac{200\text{Hz}}{5} = 40\text{Hz} \quad (6.1)$$

where f_s is the sampling frequency.

Feedback and request data from the local nodes are non-periodic. However, for ease of calculation, we approximate it as periodic. An experiment has been conducted to record the data transmitted for the local nodes over a 10 minutes period. The data transmission frequency f_l is then approximated by taking the average data transmission frequency in that period.

The number of messages per transmission for the central node n_c is equal to the number of local nodes, while n_l for the local nodes is one message at each transmission. Therefore, the throughput of each node can be calculated based on the data transmission frequency, data size (number of messages per transmission) and the message size of the periodic data as follows:

$$\text{Throughput}_{\text{central}} = n_c \times m_c \times f_c \quad (6.2)$$

$$\text{Throughput}_{\text{central}} = n_l \times m_l \times f_l \quad (6.3)$$

The total throughput is then the sum of the throughput from each node and can be calculated as follows:

$$\text{Throughput}_{\text{Overall}} = \text{Throughput}_{\text{central}} + \text{Throughput}_{\text{local}} \times \text{number of local nodes} \quad (6.4)$$

Based on (6.2), (6.3) and (6.4), the throughputs and utilization rate of the 2-fountain-head prototype and a 100-fountain-head system are calculated and listed in Table 6-1. Two of the standard CAN bit rates namely 500kbps and 1Mbps are used to calculate the utilization rate. It is noticed that the utilization rates for a 100-fountain-head system are 67.3% and 33.6% at 500kbps and 1Mbps bit rate respectively. This utilization rate is acceptable as it is significantly smaller than 100%.

To further verify that this utilization rate can be supported by the CAN bus, an experiment has been conducted. The CAN bus bit rate is selected at 500kbps in this experiment. The 2-fountain-head prototype is still used in the experimental setup. However, extra traffics have been added to the CAN bus by transmitting redundant data both from the central and the local controllers to simulate the traffic of the systems with more nodes. By adding different numbers of redundant data, different levels of throughput on the CAN bus can be created. Based on (6.2), (6.3) and (6.4), the total throughputs and utilization rates are calculated and summarized in Table 6-2. As shown in the table, by changing the data size of each nodes, utilization rate from around 50% to 95% are generated.

Number of local nodes	Sent from central controller				Sent from the local controllers				Total Throughput (kbps)	Utilization rate at 500Kbps bit rate (%)
	Message size m_c (bit)	Data frequency f_c (Hz)	Number of messages per transmission n_c	Through-put (kbps))	Message size m_l (bit)	Data frequency f_l (Hz)	Number of messages per transmission n_l	Through-put (kbps)		
2	84	40	2	6.72	44	0.1	1	0.0088	6.73	1.35
80	84	40	80	268.8	44	0.1	1	0.44	269.2	53.8
100	84	40	100	336	44	0.1	1	0.44	336.4	67.3
110	84	40	110	369.6	44	0.1	1	0.44	370.0	74.0
120	84	40	120	403.2	44	0.1	1	0.44	403.6	80.7
130	84	40	130	436.8	44	0.1	1	0.44	437.2	87.4
140	84	40	140	470.4	44	0.1	1	0.44	470.8	94.1

Table 6-2 Comparison of throughputs with different nodes

To determine whether these total throughputs can be supported by the CAN bus, we use the *local node requests* as the qualitative measure. The *local node requests* refers to the request signals transmitted by the local controllers, which includes retransmission requests, and transmission rate adjustment requests. These requests happen when transmission errors and delays occurred. Hence, a larger number of these requests indicate poor network performance since it implies more transmission error, wrong message sequence or message time delay.

Number of local nodes	2	80	100	110	120	130	140
Utilization Rate (%)	1.35	53.8	67.3	74	80.7	87.4	94.1
Number of Local node requests in 10min	86	98	89	102	121	523	Buffer empty occurred at 0.3 min

Table 6-3 Comparison of requests for different nodes

In the experiment, the number of requests received in 10 minutes at each throughput level is recorded and listed in Table 6-3. It is observed that if the utilization rate is below 80%, the network performance remains good. If the throughput is raised above 87%, there will be a significant increase in the errors. This indicates that the throughput has exceeded the maximum level. Therefore, the network performance of a system with 100 nodes is expected to be good, as the throughput of the system at 500kbps is only 67.3%.

6.6 Experimental Results with Water Spray

Due to the restriction of time and equipment, the servo motors used in the prototype are not waterproofed. As a result, all the experiments are conducted under dry condition. To validate that the proposed system functions well when water was

sprayed, a temporary waterproofing measure is used by putting plastic bags over the fountain head assembly, leaving only the nozzle outside as shown in Fig. 6-15.

In the experimental setup, the fountain head is equipped with a 6 mm diameter nozzle. A 16 Watt, 14 L/min pump is used for each fountain head to supply water. The rest of the settings remain the same. In this setup, the load has been changed due to the added water hose connection and the water running inside the nozzle. The PID gains are fine tuned in accordance to the load change.

Fig. 6-15 Experimental setup with temporary waterproofing measures

Experiments with the proposed control approach have been conducted based on this setup. Several trajectories including circular, linear and dumbbell shape trajectories are used in the experiments. The average speeds of these trajectories are 0.16 m/s, which match the typical highest speed needed for fountain applications. Since this temporary setup can not last for long, no experimental data has been recorded. However, a video clip of the experiment has been recorded. Some snapshots of the experiments are shown in Fig. 6-16.

Fig. 6-16 Snap shots of the experiments with water spay

A few observations can be made from the experiments. The experimental result shows that the nozzle moves smoothly while water is being sprayed out. There is no noticeable vibration being observed on the nozzle or water. The contouring error is also small and un-noticeable. From the above observation, it is shown that the controller can tolerate the loading effect due to water spray and there is no significant deviation in system performance.

From the fountain application's point of view, the water jets formed by the fountain heads are stable. Different water jet effects can be formed by using different

trajectories. Hence, the performance shown in the experiments is satisfactory for the intended application.

6.7 Conclusions

In this chapter, the prototype development is described. The major parts of the system including the mechanical design, the local motion controller and the central controller are presented. Implementation issues on realizing the proposed algorithms and protocol through firmware and software have also been discussed in this chapter. The prototype has been successfully constructed and used to verify the proposed control algorithms and network protocols.

Chapter 7

Conclusions and Recommendations

7.1 Conclusions

The design of a reliable network based motion control system has been presented in this thesis. It is developed for the application in fountain industry as a large scale fountain system. The system includes two major parts, the rotary biaxial motion system and the network system.

A novel cross-coupled control algorithm for rotary type biaxial mechanism has been proposed. It has been verified through simulation and experimental results that the proposed algorithm is capable of improving the contouring accuracy of the rotary biaxial motion, as compared to the conventional and direct approximation methods.

A continuous clock synchronization method has been proposed. This method is capable of generating continuous synchronized clock, which cannot be achieved by the conventional methods. By integrating the clock state and clock rate synchronization, the precision of the synchronized clock can be further improved. Analysis and experimental results have verified the performance.

A network system for synchronous motion control has been proposed based on CAN. It adopts a hierarchical structure with the central controller and local motion controller connected through the network. The application layer network protocols, including the message encoding, decoding and scheduling, motion data streaming and motion

synchronization protocols have been developed to provide reliable and effective communication channel for the network based motion control.

The biaxial motion system and the network system are integrated together to form a complete system. A prototype has been constructed to verify the system design. The proposed motion control algorithm, network protocol and clock synchronization methods have been successfully implemented in the prototype.

By incorporating this network based motion control system into a large scale fountain, greater flexibility and reliability can be achieved. The fountain nozzles placed at the end-effector of the biaxial motion mechanism have higher degree of freedom as compared to the conventional ones. Consequently, complex fountain patterns can be designed based on this type of nozzle.

On the other hand, the use of CAN based network system with the proposed application layer network protocol can greatly improve the reliability of the system. The network system not only guarantees the reliable delivering of data for motion control, but also provides a reliable channel to coordinate and monitor the whole system. Through implementation of the motion synchronization protocol, the motion of each nozzle can be precisely synchronized. Furthermore, the status data of each biaxial motion system can be sent back to the central controller for monitoring purpose. The multi-master design of the network system also provides higher reliability by using redundant central controllers.

7.2 Recommendations for Future Work

The prototype built in this project is meant for concept demonstration. For commercial product, water-proof motors and connectors are required in the real application, as the fountain nozzle will be placed underwater. Another issue in the real application is the

impact of the water spray to the biaxial motion system. The water spray from the nozzle may cause vibration. Different water pressure and spray types, e.g. laminar or turbulent, may result in different vibration strength. It is thus worthwhile to study how the water spray will affect the performance of the fountain and how it could be minimized.

For the CCC controller design, we used PID controller for both the normal and tangential axis. This controller has provided the necessary accuracy for the fountain application. However, if higher accuracy is required in other applications, it is then desirable to integrate the CCC with other advanced control techniques such as feed-forward and friction compensation.

Reference

- [1] Y. Koren, "Cross-coupled biaxial computer control for manufacturing systems," *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 102, pp. 256-272, 1980.
- [2] C. C. Lo, "A tool-path control scheme for five-axis machine tools," *International Journal of Machine Tools & Manufacture*, vol. 42, pp. 79-88, 2002.
- [3] C. LO, "Three-axis contouring control based on a trajectory coordinate basis," *JSME international journal. Series C, Mechanical systems, machine elements and manufacturing*, vol. 41, pp. 242-247, 1998.
- [4] M. Tomizuka, "Zero phase error tracking algorithm for digital control.," *J. DYN. SYST. MEAS. CONTROL.*, vol. 109, pp. 65-68, 1987.
- [5] A.Poo, "Dynamic error in type contouring systems," *IEEE Transactions on Industry Application*, vol. IA-8, pp. 477-484, 1972.
- [6] G. McKinnon, M. King, and D. Runnings, "Coordinated control of multi-axis tasks," *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, vol. 4, 1987.
- [7] I. Boldea, A. Boldea, and S. Nasar, *Electric Drives*: CRC Press, 1998.
- [8] H. Lee and M. Tomizuka, "Robust motion controller design for high-accuracy positioningsystems," *Industrial Electronics, IEEE Transactions on*, vol. 43, pp. 48-55, 1996.
- [9] K. Ohnishi, "A new servo method in mechatronics," *Trans. Jpn. Soc. Elect. Eng*, vol. 107, pp. 83?6, 1987.
- [10] K. Low and H. Zhuang, "Robust model predictive control and observer for direct driveapplications," *Power Electronics, IEEE Transactions on*, vol. 15, pp. 1018-1028, 2000.
- [11] Y. Koren and C. Lo, "Variable-gain cross-coupling controller for contouring," *Annals of the CIRP*, vol. 40, pp. 371-374, 1991.
- [12] Y. Koren and C. Lo, "Advanced controllers for feed drives.," *CIRP ANN.*, vol. 41, pp. 689-698, 1992.
- [13] G. CHIU and M. TOMIZUKA, "Coordinated position control of multi-axis mechanical systems," *Journal of dynamic systems, measurement, and control*, vol. 120, pp. 389-393, 1998.
- [14] S. Yeh and P. Hsu, "Estimation of the contouring error vector for the cross-coupledcontrol design," *Mechatronics, IEEE/ASME Transactions on*, vol. 7, pp. 44-51, 2002.

- [15] S. Chen, H. Liu, and S. Ting, "Contouring Control of Biaxial Systems Based on Polar Coordinates," *IEEE/ASME TRANSACTIONS ON MECHATRONICS*, vol. 7, pp. 329, 2002.
- [16] Y. Shih, C. Chen, and A. Lee, "A novel cross-coupling control design for Bi-axis motion," *International Journal of Machine Tools and Manufacture*, vol. 42, pp. 1539-1548, 2002.
- [17] R. FANG and J. CHEN, "Cross-Coupling Control for a Direct-Drive Robot," *JSMIE International Journal Series C*, vol. 45, pp. 749-757, 2002.
- [18] G. Chiu and M. Tomizuka, "Contouring control of machine tool feed drive systems: a taskcoordinate frame approach," *Control Systems Technology, IEEE Transactions on*, vol. 9, pp. 130-139, 2001.
- [19] G. Chiu, "Contour tracking of machine tool feed drive systems," *Proceeding of the American Control Conference Philadelphia*, pp. 3833-3837, 1998.
- [20] S. Yeh and P. Hsu, "Analysis and design of integrated control for multi-axis motion systems," *Control Systems Technology, IEEE Transactions on*, vol. 11, pp. 375-382, 2003.
- [21] S. Yeh and P. Hsu, "Analysis and design of the integrated controller for precise motionsystems," *Control Systems Technology, IEEE Transactions on*, vol. 7, pp. 706-717, 1999.
- [22] L. YingChen, "Precise motion control for multi-axis system using real time forecast cross-coupling controller," *Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th*, vol. 3, 2004.
- [23] M. Yan, M. Lee, and P. Yen, "Theory and application of a combined self-tuning adaptive control and cross-coupling control in a retrofit milling machine," *Mechatronics*, vol. 15, pp. 193-211, 2005.
- [24] P. Kulkarni and K. Srinivasan, "Optimal Contouring Control of Multi-Axial Feed Drive Servomechanisms," *ASME Journal of Engineering for Industry*, vol. 111, pp. 140-148, 1989.
- [25] Y. XIAO, K. ZHU, and H. LIAW, "Generalized synchronization control of multi-axis motion systems," *Control engineering practice*, vol. 13, pp. 809-819, 2005.
- [26] Z. Yeh, "A cross-coupled bistage fuzzy controller for biaxis servomechanism control," *Fuzzy Sets and Systems*, vol. 97, pp. 265-275, 1998.
- [27] J. Chin, Y. Cheng, and J. Lin, "Improving contour accuracy by Fuzzy-logic enhanced cross-coupled precompensation method," *Robotics and Computer Integrated Manufacturing*, vol. 20, pp. 65-76, 2004.
- [28] S. Yeh and P. Hsu, "Theory and applications of the robust cross-coupled control design," *American Control Conference, 1997. Proceedings of the 1997*, vol. 1, 1997.
- [29] G. Collins, "The foundations of celestial mechanics," *Tucson, AZ, Pachart Publishing House (Pachart Astronomy and Astrophysics Series.*, vol. 16, pp. 158, 1989.

- [30] F. Lian, J. Moyne, and D. Tilbury, "Performance evaluation of control networks- Ethernet, ControlNet, and DeviceNet," *IEEE Control Systems Magazine*, vol. 21, pp. 66-83, 2001.
- [31] R. B. GmbH, *CAN Specification Version 2.0*: GmbH, Robert Bosch.
- [32] ISO, "11519-1 Road Vehicles-Low Speed serial data communication-Part 1: Low Speed Controller Area Network," ISO. ISO, 1994.
- [33] ISO, "11519-2 Road Vehicles-Low Speed serial data communication-Part 2: Low Speed Controller Area Network," ISO. ISO, 1994.
- [34] N. Xiaodong, Z. Yanjun, M. Departunent, V. Technoloizies, and C. Beijing, "Determining message delivery delay of controller area networks," *TENCON'02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, vol. 2, 2002.
- [35] J. Charzinski, "Performance of the error detection mechanisms in CAN," *Proceedings of the 1st International CAN Conference*, pp. 1.20?.29, 1994.
- [36] G. Walsh and H. Ye, "Scheduling of networked control systems," *Control Systems Magazine, IEEE*, vol. 21, pp. 57-65, 2001.
- [37] M. Chow and Y. Tipsuwan, "Gain adaptation of networked DC motor controllers based on QoS variations," *Industrial Electronics, IEEE Transactions on*, vol. 50, pp. 936-943, 2003.
- [38] M. Livani, J. Kaiser, and W. Jia, "Scheduling hard and soft real-time communication in a controller area network," *Control Engineering Practice*, vol. 7, pp. 1515?523, 1999.
- [39] K. Zuberi and K. Shin, "Non-Preemptive Scheduling of Messages on Controller Area Network for Real-Time Control Applications," *Proc. Real-Time Technology and Applications Symposium*, pp. 240?49, 1995.
- [40] M. Di Natale, "Scheduling the CAN bus with earliest deadline techniques," *Real-Time Systems Symposium, 2000. Proceedings. The 21st IEEE*, pp. 259-268, 2000.
- [41] D. H. G Leen, "TTCAN: a new time-triggered controller area network," *Microprocessors and Microsystems*, 2002.
- [42] L. Almeida, P. Pedreiras, and J. Fonseca, "The FTT-CAN Protocol: Why and How," *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, vol. 49, pp. 1189, 2002.
- [43] H. Boterenbrood, "CANopen, high-level protocol for CAN-bus," *NIKHEF, Amsterdam, Version3*, vol. 3.
- [44] C. Standard, "凖 AN Application Layer ?" *CiA/DS*, pp. 201-207, 1994.
- [45] U. Schmid and K. Schossmaier, "Interval-based Clock Synchronization," *Real-Time Systems*, vol. 12, pp. 173-228, 1997.
- [46] D. Mills, "Network Time Protocol (Version 3) Specification, Implementation and Analysis," *Network*, 1992.

- [47] R. Gusella and S. Zatti, "The accuracy of the clock synchronization achieved by TEMPO in Berkeley UNIX 4.3 BSD," *Software Engineering, IEEE Transactions on*, vol. 15, pp. 847-853, 1989.
- [48] P. Ramanathan, K. Shin, and R. Butler, "Fault-tolerant clock synchronization in distributed systems," *Computer*, vol. 23, pp. 33-42, 1990.
- [49] J. Welch and N. Lynch, "A new fault-tolerant algorithm for clock synchronization," *Information and Computation*, vol. 77, pp. 1-36, 1988.
- [50] L. Lamport and P. Melliar-Smith, "Synchronizing clocks in the presence of faults," *Journal of the ACM*, vol. 32, pp. 52-78, 1985.
- [51] T. Srikanth and S. Toueg, "Optimal clock synchronization," *Journal of the ACM (JACM)*, vol. 34, pp. 626-645, 1987.
- [52] D. Dolev, J. Halpern, B. Simons, and R. Strong, "Dynamic fault-tolerant clock synchronization," *Journal of the ACM (JACM)*, vol. 42, pp. 143-185, 1995.
- [53] P. Verissimo and L. Rodrigues, "A posteriori agreement for fault-tolerant clock synchronization on broadcast networks," *Proc. of 22nd International Symposium on Fault-Tolerant Computing Systems*, 1992.
- [54] L. Rodrigues, M. Guimaraes, and J. Rufino, "Fault-Tolerant Clock Synchronization in CAN," *Proceedings of the 19th Real-Time Systems Symposium*, pp. 420-429, 1998.
- [55] H. Kopetz, A. Ademaj, and A. Hanzlik, "Integration of internal and external clock synchronization by the combination of clock-state and clock-rate correction in fault-tolerant distributed systems," *Real-Time Systems Symposium, 2004. Proceedings. 25th IEEE International*, pp. 415-425, 2004.
- [56] K. Schossmaier and B. Weiss, "An Algorithm for Fault-Tolerant Clock State&Rate Synchronization," *Proceedings 18th IEEE Symposium on Reliable Distributed Systems (SRDS'99)*, pp. 36-47, 1999.
- [57] K. Schossmaier, "An interval-based framework for clock rate synchronization," *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pp. 169-178, 1997.
- [58] P. Ramanathan, D. Kandlur, and K. Shin, "Hardware-assisted software clock synchronization for homogeneous distributed systems," *IEEE Transactions on Computers*, vol. 39, pp. 514-524, 1990.
- [59] M. Mock, R. Frings, E. Nett, and S. Trikaliotis, "Continuous clock synchronization in wireless real-time applications," *The 19th IEEE Symposium on Reliable Distributed Systems (SRDS'00)*, pp. 125-133, 2000.
- [60] F. Schmuck and F. Cristian, "Continuous clock amortization need not affect the precision of a clock synchronization algorithm," *Proceedings of the ninth annual ACM symposium on Principles of distributed computing*, pp. 133-143, 1990.
- [61] H. Kopetz, A. Kruger, D. Millinger, and A. Schedl, "A synchronization strategy for a time-triggered multicluster real-time system," *Proc. of the 14th IEEE Symposium on Reliable Distributed Systems, Bad Neuenahr, Germany*, vol. 1315, 1995.

- [62] "Fuji Servo System FALDIC-W Series User's Manual," Fuji Electric FA Components & System Co., Ltd., 03 2004.
- [63] Microchip, "Microchip PIC18F6585/8585/6680/8680 Data Sheet," Microchip Technology Inc., 2004.
- [64] "Adapter Card PC-PCI to High-speed CAN User Manual," PEAK-System Technik GmbH, May 3 2006.
- [65] G. CRAESSAERTS, K. MAERTENS, and J. DE BAERDEMAEKER, "A windows-based design environment for combine automation via CAN bus," *Computers and electronics in agriculture*, vol. 49, pp. 233-245, 2005.

Appendices

A. Data Types of the Network Based Motion Control Application

Data types are defined for the network based motion control application. The first byte of CAN message data field is defined as the header of that message, which specifies the data type. All the data types and their corresponding header values are listed in the table below.

Header value	Sender - receiver	Data class	Description
87	Local to central	status	Local node is working
83	Local to central	status	Local node is on Standby
85	Local to central	header	CCC error data
82	Local to central	header	Tracking error data
79	Local to central	status	Power On
69	Local to central	status	Unrecognized command received
78	Local to central	header	Position feedback data
80	Local to central	status	P mode selected
76	Local to central	status	Open loop mode selected
73	Local to central	status	PI mode selected
68	Local to central	status	PID mode selected
67	Local to central	status	CCC mode selected
72	Local to central	status	“encoder reading feedback” selected
66	Local to central	status	“error reading feedback” selected
71	Local to central	status	Upload mode selected
77	Local to central	status	Upload completed
81	Local to central	status	board control Mode selected
88	Local to central	status	Motion tracking completed
86	Local to central	status	PC control mode selected
90	Local to central	status	Board control loop mode selected
105	Local to central	status	Error: start up check failed
106	Local to central	status	Startup check completed
109	Local to central	status	Parameter mode selected
110	Local to central	status	parameter complete
111	Local to central	status	CAN mode selected
112	Local to central	status	USART mode selected
114	Local to central	request	Request to send data slower
117	Local to central	request	Request to send data faster
120	Local to central	status	Buffer error, motion forced stop
122	Local to central	request	Request to retransmitted position data
33	Central to local	command	Start motion

34	Central to local	command	Start with delay
35	Central to local	command	Stop motion
39	Central to local	command	Request to update current status
40	Central to local	header	Position data header
42	Central to local	command	Request for current encoder reading
45	Central to local	command	Request for sending encoder readings continuously
51	Central to local	command	Change control mode to Open Loop
54	Central to local	command	Request for sending tracking error readings continuously
57	Central to local	command	Stop sending encoder readings continuously
60	Central to local	command	Stop sending tracking error readings continuously
63	Central to local	command	Request for sending CCC error readings continuously
66	Central to local	command	Stop sending CCC error readings continuously
69	Central to local	command	Change control mode to P mode
72	Central to local	command	Change operating mode to Upload mode
75	Central to local	command	Change operating mode to PC control Mode
78	Central to local	command	Change operating mode to Board control Mode
81	Central to local	command	Change operating mode to normal mode
84	Central to local	command	Loop on the board control mode
87	Central to local	command	No loop on the board control mode
91	Central to local	command	Change control mode to PID mode
94	Central to local	command	Change control mode to CCC mode
95	Central to local	command	Poll parameters
98	Central to local	command	Parameters update start
101	Central to local	command	Parameters update mode completed
102	Central to local	command	CCC mode CCC Proposed
103	Central to local	command	CCC mode CCC Direct
104	Central to local	command	CCC mode P
105	Central to local	command	Turn on pre start option
106	Central to local	command	Turn off pre start option
107	Central to local	command	Turn on pre stop option
108	Central to local	command	Turn off pre stop option
109		command	Turn on interpolation option
110	Central to local	command	Turn off interpolation option
111	Central to local	command	Perform startup check
114	Central to local	command	Use pre loaded map
115	Central to local	command	Use the just loaded map
116	Central to local	header	Header for map selection data
117	Central to local	command	CAN communication mode
118	Central to local	command	USART communication mode
120	Central to local	command	Nozzle go back to origin
121	Central to local	command	Nozzle go back to center

123	Central to local	command	x high byte is 0(only used in rs232)
124	Central to local	command	y high byte is 0(only used in rs232)
125	Central to local	command	both x, y high byte are 0(only used in rs232)
126	Central to local	command	Use Buffer ON
127	Central to local	command	Do not use Buffer
133	Central to local	command	Reset the controller board

B. Schematic Diagram of the Controller Board

B1. Encoder Interface Schematic

B2. Servo Amplifier Interface Schematic

B3. LED Display Schematic

B4. Microcontroller I/O Connection Schematic

B5. Power Regulation Unit Schematic

B6. Communication Driver Schematic