Implementation of Decentralized Formation Control on Multi-Quadrotor Systems

by

Nasrettin Koksal

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical Engineering

Waterloo, Ontario, Canada, 2014

Abstract

We present real-time autonomous implementations of a practical distributed formation control scheme for a multi-quadrotor system for two different cases: parameters of linearized dynamics are exactly known, and uncertain system parameters. For first case, we design a hierarchical, decentralized controller based on the leader-follower formation approach to control a multi-quadrotor swarm in rigid formation motion. The proposed control approach has a two-level structure: high-level and low-level. At the high level, a distributed control scheme is designed with respect to the relative and global position information of the quadrotor vehicles. In the low-level, we analyze each single quadrotor control design in three parts. The first is a linear quadratic controller for the pitch and roll dynamics of quadrotors. The second is proportional controller for the yaw motion. The third is proportional-integral-derivative controller in altitude model. For the second case, where inertial uncertainties in the pitch and roll dynamics of quadrotors are considered, we design an on-line parameter estimation with the least squares approach, keeping the yaw, altitude and the high-level controllers the same as the first case. An adaptive linear quadratic controller is then designed to be used with lookup table based on the estimation of uncertain parameters. Additionally, we study on enhancement of self and inter-agent relative localization of the quadrotor agents using a single-view distance-estimation based localization methodology as a practical and inexpensive tool to be used in indoor environments for future works. Throughout the formation control implementations, the controllers successfully satisfy the objective of formation maintenance for non-adaptive and adaptive cases. Simulations and experimental results are presented considering various scenarios, and positive results obtained for the effectiveness of our algorithm.

Acknowledgements

I would like to thank my supervisor, Dr. Baris Fidan for the guidance and support throughout my graduate studies.

I would also like to thank Samet Guler, Kemal Buyukkabasakal, and Elshad Mustafayev. Their assistance was countless and extremely important.

Also, I would like to thank to Dr. William W. Melek and Dr. Nasser Lashgarian Azad for being my readers.

I am also grateful to the National Education Ministry of Turkey for financial support.

Finally, to my family, I wouldn't have made it without your faith and support in me.

To my family

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and Earlier Works

Recently, based on robotic development, performance requirements of robotic systems have been increased for difficult missions and environment. Since most single robotic systems are not efficient for work in some of complicated team tasks such as military patrol duties, agricultural activities, search, surveillance, rescue etc., multi-agent dynamic systems are more useful than single robotic systems. Furthermore, multi-agent dynamic system design and analysis has become an interdisciplinary subject, having applications in various fields involving aerial, submarine, satellite and ground systems [1–7]. Considering multi-agent systems, cooperative and cohesive motion tasks have been a relatively new field of research in recent decades. Designing control schemes for these tasks have some difficulties as a consequence of complexities and components of multi-dynamic structures. Typical control tasks for such cooperative dynamic systems include path planing, flocking, consensus, obstacle avoidance, collision, stability analysis, formation problems [2, 5, 8–15]. Such tasks have prospective applications in various field, including cooperative defence robotics, cooperative surveillance, cooperative source localization and tracking.

In the motion of a multi-agent system as a *swarm*, formation is used for a swarm collection to perform certain cooperative mission requirements, optimally [16]. In other words, the formation control task aims to structure a swarm system in a fixed pattern while a system is moving on a desired trajectory. In order to solve the formation problems, several architectures have been considered hierarchical vs. non-hierarchical and symmetric vs. asymmetric [1,2,5,17]. Note that hierarchical control structure is more practical because of its applicability to the systems without long-range sensors except in case of leader agents. Non-hierarchical structures, on the other hand, use a virtual leader approach [12, 13, 17]. The fictitious virtual leader is used as a reference path generator for a fictitious point (e.g. geometric center) of the formation. All agents are assumed to be responsible for maintaining inter-agent distances between the virtual leader and each follower with non-hierarchical architecture. In this thesis, using a leader-follower formation approach, we perform an asymmetric decentralized control design for a multi-quadrotor system which is designated to move in an area maintaining a predefined formation. To make the leader agent more free when it makes decisions, we use hierarchical control structure as studied in [1, 2].

Later, we consider various parametric uncertainties in the system and design an adaptive version of our distributed control scheme. In literature, for adaptive control design, there are two main approaches: direct and indirect adaptive control approaches [18]. Indirect method calculates controller parameters using estimated system parameter at each instant time, while direct method updates directly controller parameters. Considering the system inertial uncertainties in pitch and roll dynamics, we also perform the asymmetric decentralized control design using an indirect adaptive control approach based on the least square estimation algorithm.

With respect to multi-quadrotor system, quadrotor helicopters have been discussed in many works of literature such as [19–24] where there are some of important considerations about designing quadrotor dynamics, namely, external disturbance effect on the system dynamics, blade flapping, torque effects from propellers, inertial effects from center of mass, fault detection and tolerance, and trajectory generation. In this thesis, we use a dynamic model of a multi-quadrotor system from [22, 25] to design overall control schemes for non-adaptive and adaptive cases. As in [15], we consider the overall control task of the both cases in two parts namely high level and low-level control structures. At first, the controller is related with the formation maintaining and generation of desired attitude angles. Second, the controller is responsible for controlling an individual agent. We also use a linear quadratic controller (LQR), proportional (P), and propositional-integral-derivative (PID) controller for pitch and roll, yaw, and altitude model, respectively. For the adaptive control scheme, we design an adaptive linear quadratic controller (ALQR). In both cases, our main mission is to maintain required distances, positions or specified orientation in a swarm instead of individual agent behaviors. To better illustrate the formation behaviors using the proposed control schemes, we fix a set of specific formation missions in upcoming chapters. In order to perform real-time experimental testing, we use a group of Qball-X4 quadrotor vehicles designed by Quanser Inc.. During formation missions, although sensor information provides some variables for the quadrotor such as acceleration and rotation angles by the inertial measurement unit (IMU), we use an external localization system to make more accurate position information. Because of our chosen indoor environment for test, GPS (Global Positioning System) is not feasible for obtaining positions of swarm agents on our system. Therefore, we use an indoor localization method, namely Optitrack system designed by NaturalPoint Inc., integrated to our multi-quadrotor testbed.

1.2 Contributions and Organization of the Thesis

Designing motion controller for a single-quadrotor agent or a multi-quadrotor system carries many difficulties due to dynamic nonlinearities and uncertainties, and mission complexities as discussed in earlier literature [3, 19–23]. In this study, by consideration of these works, we first focus on real-time implementation of autonomous formation control of Qball-X4 multi-quadrotor system with the leader-follower structure. By means of the tools provided in [1, 3, 4, 17], we develop a distributed formation acquisition and maintenance control algorithm based on measurements of inter-agent distances. The control algorithm establishes a maintained geometric configuration during the swarm motion. Orientations and global positions of the quadrotors are provided by inertial measurement units (IMU) and the Optitrack optical sensors in indoor environment are used as feedback signals. Then, we consider uncertainties in the Qball-X4 quadrotor dynamic parameters, particularly in inertia of pitch and roll dynamics for the same formation task. We develop an adaptive control scheme based on the least squares on-line parameter identification (PI). Finally, we focus on enhancement of the localization tools used in the distributed formation control, and propose a localization enhancement methodology based on use of a single camera mounted on each quadorotor.

In summary the main contributions of this thesis are as follows:

(i) we design and analyze a practical distributed formation control scheme for real-time implementation of rigid formation maintenance tasks on multi-quadrotor systems.

(ii) In order overcome inertial uncertainties during the motion, we design an adaptive version of the developed practical distributed formation control scheme.

(iii)Utilizing single-view based position estimation for each quadrotor, we design an optimal localization enhancement method for better accuracy and longer operation field.

The rest of the thesis is organized as follows: Background and the modeling of the quadrotor system are presented in Chapter 2. The base real-time formation control architecture is designed and implemented in Chapter 3. The adaptive control version of this control architecture is designed and tested Section 4. In Chapter 5, we design a method for localization enhancement. Concluding remarks are given in Chapter 6.

Chapter 2

Background

2.1 Graph Modeling of Vehicle Networks

In this section, we give essential definitions about graph theoretical modeling of multi-agent vehicle networks (swarms) in order to construct a required formation architecture for an autonomous formation task of a multi-quadrotor system. In the formation control literature [26–28], graph theory is a significant tool used for geometric representation of a group of nodes, which represent agents of a swarm S for a formation task, in a specified motion space, e.g. $\mathbb{R}^2$ or $\mathbb{R}^3$. Thus, a graph is defined as a theoretical interconnection structure among agents to represent information flow, communication and interaction topology within the formation structure of a swarm S. Moreover, graph topology is not only used for a required mission such as moving into formation, maintenance of formation form or switching between formations, but also it is used to analyze their controllability and stability. Formally, a graph G is a pair $G = (V, E)$ which consists of a vertex set V, and an edge set $E \subset V \times V$; each vertex represents an agent and communication sensing and each edge connecting two vertices represents a link between corresponding agents. For example, in Figure 2.1-[a], vertices $V = (1, 2, 3)$ as agents $A_i, \quad i = 1, 2, 3$ and edges $E = (\overrightarrow{13}, \overrightarrow{32}, \overrightarrow{21})$ as directed information links. In addition, when inter-agent distance control is considered, there are two types of approaches for controlling these inter-agent distances. In terms of graph types, they are called symmetric or asymmetric structures [26, 28–30]. These control structures can be represented by directed or indirected graphs, respectively, as seen in Figure 2.1.

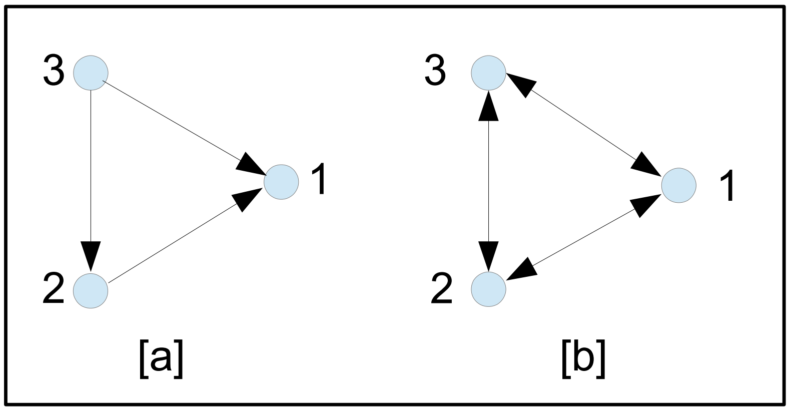


Figure 2.1: Graph types for a three-agent formation control [a] Directed graph [b] Undirected graph

2.1.1 Directed Graph Modeling

Considering the asymmetric formation control structure, in order to generate a directed underlying graph $G_F = (V_F, E_F)$ of a swarm, we use the directions of information links within graph G_F. If the graph G_F involves a directed link between $\overrightarrow{(i, j)}$ agents, it is called a directed underlying graph. For instance, as seen in Figure 2.1-[a], there are three ordered pairs among agents $(1, 2, 3)$, so these ordered pairs represent directed information links in connection architecture of agents. When considering the formation task in a multi-agent system, one of neighbor agent in pairs is responsible for receiving position information by using a communication network or sensing system. Using the asymmetric formation control structure, the responsible agent makes decisions by itself to maintain the desired goal. Therefore, when compared to the symmetric control structure, the multi-agent formation control design avoids half of the complexities in communication and control complexities. [29].

2.1.2 Undirected Graph Modeling

The symmetric formation control structure has undirected information links between agent pairs. In order to generate an undirected graph $G_F = (V_F, E_F)$, the properties of information links are also important within underlying graph G_F. If both agent pairs are responsible in maintenance of formation tasks, G_F is undirected underlying graph. For instance, as seen in Figure 2.1-[b], there are three undirected agent pairs, so they have undirected information links in connection architecture between each responsible agent pairs. Thus, this represents an undirected underlying graph. Considering the formation task, in the symmetric structure, since both agent pairs are responsible to maintain desired inter-agent distance, this causes twice the communication and control complexities as compared to an asymmetric control structure.

2.2 Rigid and Persistent Formations

Using graph theory tools, the formation of a swarm S is represented by $F = (S, G_F, D_F)$ where $G_F = (V_F, E_F)$ is underlying graph, $S = (A_1, ..., A_i), \quad i \geqslant 2$ represents members of swarm, and $D_F = \{d_{ij} | \overrightarrow{(i,j)} \in E_F\}$ is objective distance set. Considering the formation F and each of its agent pairs (A_i, A_j), if all distances d_{ij} are maintained in F, it is called a *rigid* formation by condition of inter-agent distances d_{ij}. In other words, by keeping inter-agent distances of each agent pairs, the desired formation shape remains at its required distances within the formation shape. In addition, during the formation task, if each agent satisfies the inter-agent distance constraints, the formation F is called a $constraint - consistent$. Furthermore, if F satisfies the *rigid* and $constraint - consistent$ formation, this is called a *persistent* formation [31, 32]. Besides, to satisfy $minimal - persistent$, the formation edges should be in the limitation which is minimal number of edges to provide persistence.

2.3 Formation Control Structures

In the formation control various methods have been developed, including leader-follower, virtual leader, co-leader and behavior-based approaches. Here, we just review the leader-follower approach and virtual leader approach to offer a brief background.

2.3.1 The Leader-Follower Approach

The leader follower approach is practical for many formation studies, such as seen leader follower tasks. [1, 3, 5], because of leader's freedom in guiding the formation and having the formation free of cyclic control paths. One other benefit of this approach is that the only leader agent can be equipped with long-range sensors for localization. Others can be equipped with unsophisticated short-range distance measurement sensors such as cameras, sonar, and infrared sensors in order to measure distances between agent pairs.

The main notion of this approach is that one agent is designed as a leader, while the others are designated to follow the leader as followers in formation structure. Therefore, the leader agent does not attempt to maintain the formation shape, and it has 3 degrees of freedom (DOF) in the motion. It tracks a prior planned or on-line decided desired trajectory in $\mathbb{R}^3$. Concurrently, the second agent aims to maintain desired distance from the leader with 2 DOF, left, to be used orienting the formation. Thereby, it is called the $first\ follower$. In a similar way, the third agent aims to maintain desired distances from the leader and the first follower, and uses the 1 DOF to help the first follower in orientation of the formation. It is called the $second\ follower$. Each of the other agents maintains pre-set distances from 3 pre-assigned neighbor agents, having 0 DOF left for free motion, and it is called $ordinary\ follower$. For $\mathbb{R}^2$-motion case, [1] and as seen in Figure 2.2, the leader has 2 DOF with independent from other agents' positions and heading angles, and it simply tracks its desired angle as a free agent. With respect to distance error threshold between the leader and second agent, the second agent follows the generated heading angle as a $first\ follower$, and its desired angles depend on the leader's position and heading angle with 1 DOF. Furthermore, the last agent depends on both agent informations to generate its desired angles with 0 DOF, and it is called $ordinary\ follower$ in motion.

2.3.2 The Virtual Leader Approach

In the multi-agent systems' formation control problems, the agents usually have limited communicating and sensing capabilities. Hence, information acquisition can be limited to

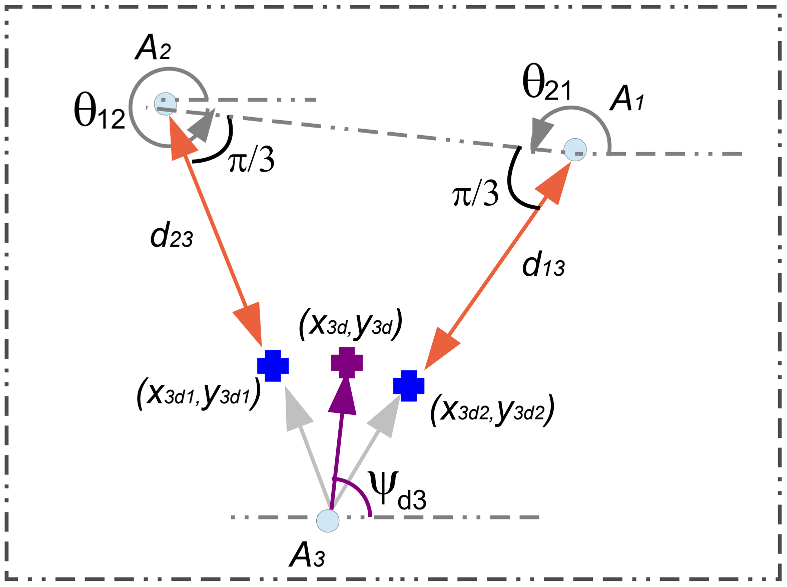


Figure 2.2: A leader-follower formation approach generating desired angles [1]

inter-agent measurements among members of a formation swarm. In the present case, the developed control laws need to rely only on local information of neighbor agents for satisfying the targets such as keeping formation, obstacle avoidance, and reference tracking [4,5]. On the other hand, in the virtual leader approach, assuming that each agent is able to sense its essential information in a perfect manner [6,17]. A virtual leader is defined in suitable place of the formation that is designated for a reference path. During the task, each agent is responsible to maintain the inter-agent distance between the virtual leader and itself in the non-hierarchical architecture. In other words, all agents are controlled, and then they position around the virtual leader on the desired position and/or bearing. In [17], one virtual agent is defined to track the desired virtual path as the leader. Then, the motion of virtual leader is adapted to desired motion for each agent. Lastly, each agent tracking is derived with respect to individual consideration as illustrated in Figure 2.3.

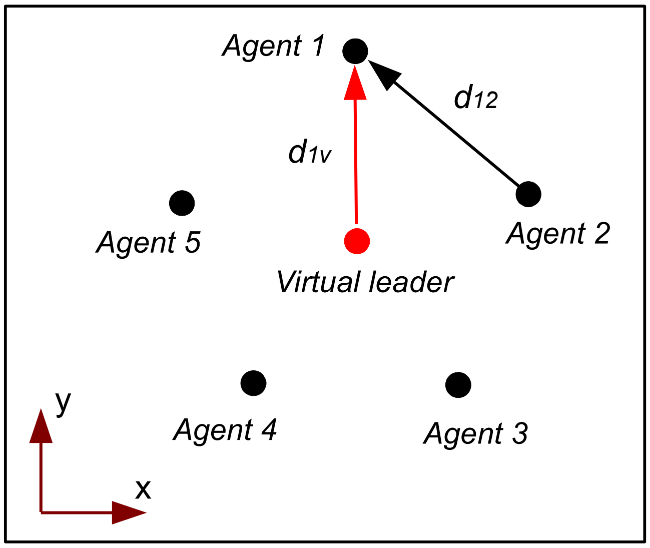


Figure 2.3: A virtual leader formation approach [17]

2.4 Multi-Quadrotor System

As the testbed for our study, we use a multi-quadrotor system [25] composed of Qball-X4 quadrotors developed by Quanser Inc., and shown in Figure 2.4. For the sake of secure operation, each Qball is enclosed by protective carbon fiber sticks to avoid hazards in indoor tests. Each Qball is equipped with two pairs of brushless DC motors and fitted 10-inch propellers. The Qball-X4 has an on-board avionics data acquisition card (DAQ) and Gumstix embedded computer, as shown in Figure 2.5 for interfacing with on-board sensors and driving the motors. The DAQ includes an inertial measurement unit (IMU) and an avionics input/output (I/O) card to support different studies on the quadrotor. To drive the motors, each motor is linked to one of the four PWM servo output channels in summarized Table 2.1 on the DAQ. Furthermore, Quanser's real time software Quarc and its MATLAB Simulink® interface provide easy developing and testing on actual hardware.

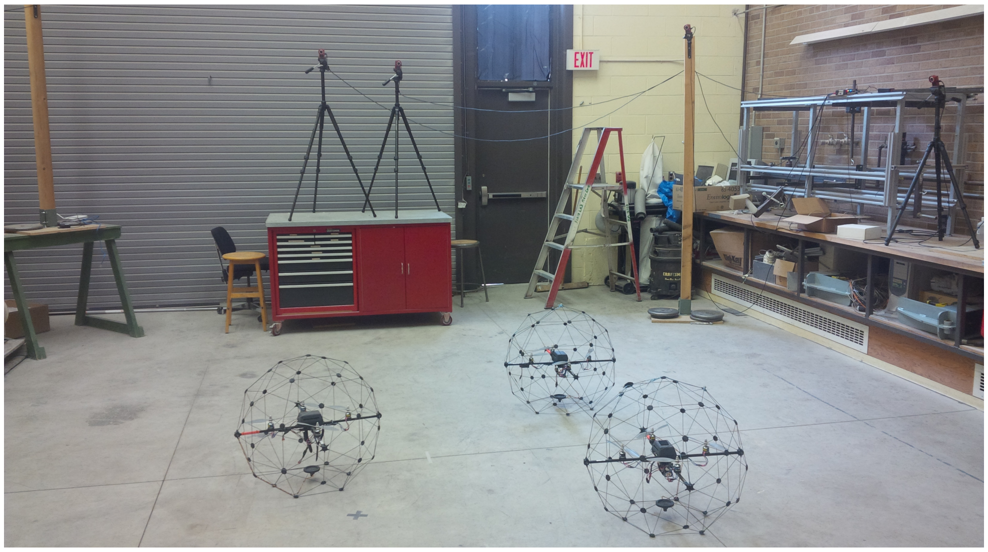


Figure 2.4: Multi-quadrotor Qball-X4 swarm system

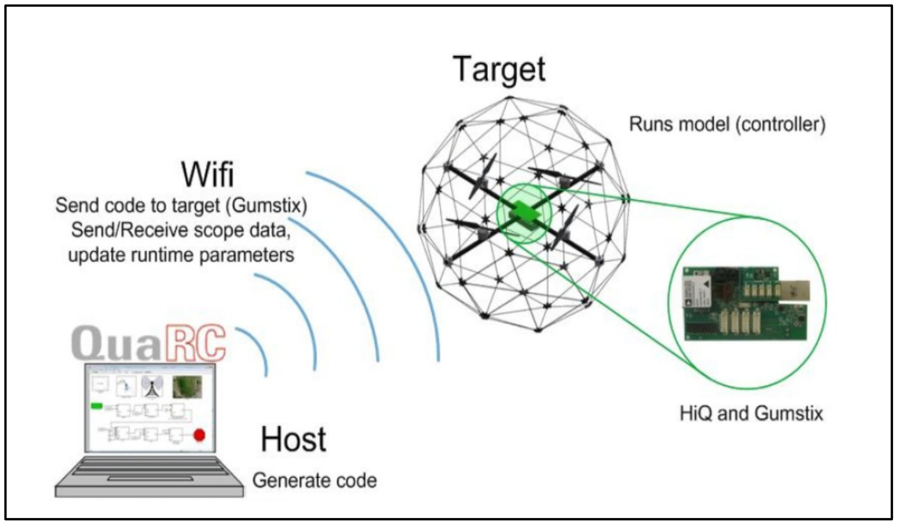


Figure 2.5: Communication of the Qball-x4 quadrotor [25]

Table 2.1: Motor servo channels [25]

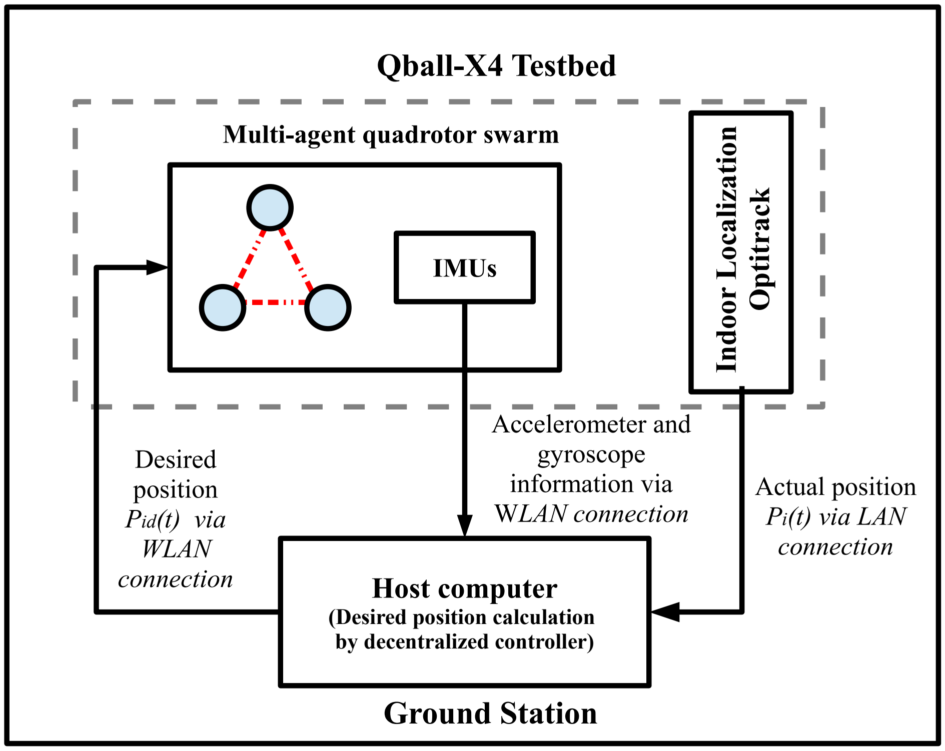
Motor	Back	Front	Left	Right
Servo output channel	0	1	2	3

2.4.1 Communication and Sensing of Multi-Quadrotor System

In our experimental study, within communication architecture, we have a multi-quadrotor testbed which includes the multi-quadrotor swarm, the Optitrack localization system and ground station as summarized in Figure 2.6. In the testbed, the host computer is in charge of communication among the members of the swarm. In addition, in order to reduce workload on the Gumstix embedded computer, the host computer is used to operate high-level controller for the generation of the agents' desired position at any time instant. At that time, the indoor localization system Optitrack helps the high-level controller by supporting the actual position information for more accurate localization. Also, inertial measurement units (IMU) provide acceleration and angular orientation measurements. Therefore,using these sensor feedback signals, the host computer generates desired position informations, and then the computer sends them to individual quadrotor motion controllers as reference inputs. Among testbed units, all communication is via WLAN connection.

2.4.2 OptiTrack Indoor Localization System

Our indoor localization system has fourteen NaturalPoint OptiTrack cameras [33] as shown in Figure 2.7. To receive position data from the localization system, we use Quarc's Optitrack block and predefined trackables of the OptiTrack software. Also, in order to receive position information of predefined trackable agents, the indoor localization system



Qball-X4 Testbed
Multi-agent quadrotor swarm
IMUs
Indoor Localization Optitrack
Accelerometer and gyroscope information via WLAN connection
Desired position Pid(t) via WLAN connection
Actual position Pi(t) via LAN connection
Host computer (Desired position calculation by decentralized controller)
Ground Station

Figure 2.6: Multi-quadrotor Qball-X4 testbed

is directly connected to the host computer by an LAN connection as shown in Figure 2.6, and markers of different shapes are placed on the quadrotor agents for sensing.

2.4.3 Coordinate Frames

In this study, our operational frames are a global frame O_g and the fixed-body frame O_b as depicted in Figure 2.8, respectively. In fact, there are more frames involved in the whole design such as camera, image, and inertial frames. We skip them in the thesis presentation for brevity. All moments, forces, and inertia are taken in fixed-body frame as indicated in

Figure 2.8, and the frames can be transformed by aid of Rotation matrix

$$R = \begin{bmatrix} cos\theta cos\psi & -cos\phi sin\psi + sin\phi sin\theta cos\psi & sin\psi sin\phi + cos\phi sin\theta cos\psi \\ cos\theta sin\psi & cos\phi cos\psi + sin\phi sin\theta sin\psi & -sin\phi cos\psi + cos\phi sin\theta sin\psi \\ -sin\theta & sin\phi cos\theta & cos\phi cos\theta \end{bmatrix} \tag{2.4.1}$$

and the corresponding Euler angles. In addition, on the ground plane of the workspace, a origin point is chosen as a global frame O_g in implementation.

Figure 2.7: OptiTrack Camera [33]

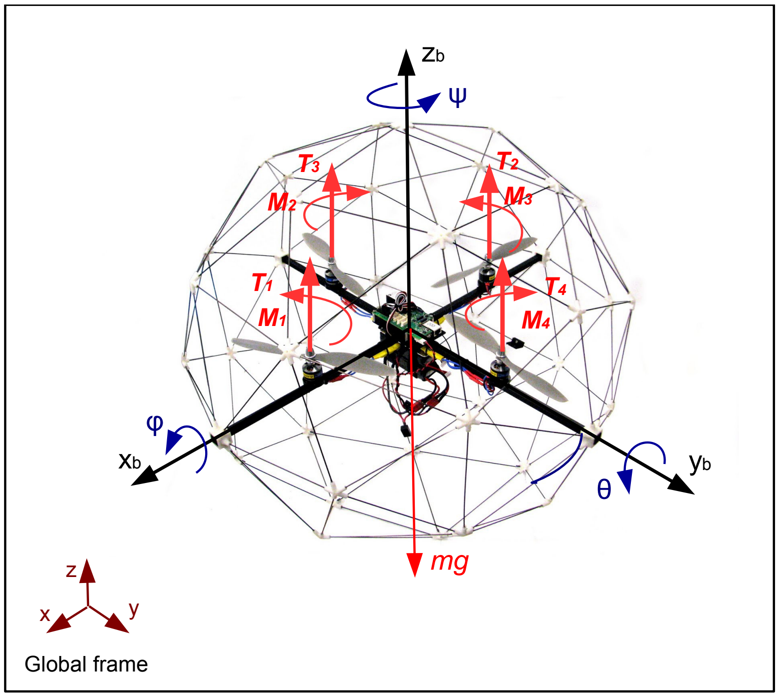


Figure 2.8: The Qball-X4 quadrotor UAV in the real-time indoor autonomous test and its coordinate representation as body frame O_b and global frame O_g with thrust, moment and gravity effects [25].

2.5 Modeling of Quadrotor Agents

This section sets out to explain how the linearized dynamic models of the QBall-X4 quadrotor system are designed [22, 25] to build an overall control scheme. Therefore, firstly, we discuss about nonlinear form of quadrotor motion dynamics and then the QBall-X4's motion equations are described as linear models for use in our control design.

2.5.1 Equations of Motion

Considering a quadrotor system, motion equations are discussed in form of nonlinear structure as in earlier works [3,19,22] with regard to different dynamics effects on the system. In our study, we use dynamic models of autonomous QBall-X4 quadrotor vehicle as presented in [22,25]. Modeling QBall-X4 quadrotor vehicle, nonlinear dynamics of the quadrotor are described in terms of applied forces $F \in \mathbb{R}^3$,

$$F = [F_x \ F_y \ F_z]^T = ma, \tag{2.5.1}$$

and moments $M \in \mathbb{R}^3$,

$$M = J\dot{w} + w \times Jw, \tag{2.5.2}$$

where m is mass of the quadrotor, J is the rotational inertia, w is the angular velocity and $a = [\ddot{x} \ \ddot{y} \ \ddot{z}]^T$ is linear acceleration in a 3D global frame.

On the body frame of the quadrotor system, the four actuators and their propellers generate four different thrust forces as shown in Figure 2.8, and all forces effect on a positive direction of z-axes as shown below:

$$F_b = \begin{bmatrix} F_{xb} \\ F_{yb} \\ F_{zb} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ T_1 + T_2 + T_3 + T_4 \end{bmatrix}. \tag{2.5.3}$$

.

Using rotation matrix (2.4.1), we define in 3D global frame as follows:

$$F = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = R \begin{bmatrix} F_{xb} \\ F_{yb} \\ F_{zb} \end{bmatrix} = \begin{bmatrix} sin\psi sin\phi + cos\phi sin\theta cos\psi \\ -sin\phi cos\psi + cos\phi sin\theta sin\psi \\ cos\phi cos\theta \end{bmatrix} \begin{bmatrix} T_1 + T_2 + T_3 + T_4 \end{bmatrix}. \tag{2.5.4}$$

.

Therefore, by Newton's second law (2.5.1) and using drag effects as discussed in [3,22], in a global frame, the equation of motion is written by adding the negative direction of the gravitational effect on the z-plane as follows:

$$a = \frac{F - f}{m}, \tag{2.5.5}$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} F_x - f_x \\ F_y - f_y \\ F_z - mg - f_z \end{bmatrix} = \frac{1}{m} \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} - \frac{1}{m} \begin{bmatrix} f_x \\ f_y \\ mg + f_z \end{bmatrix}, \tag{2.5.6}$$

where drag forces $f = [f_x,\ f_y,\ f_z]^T$ are combined within the equation of motion.

Moreover, these forces are defined in works mentioned earlier [3,22,34] as follows:

$$f = D_b v, \tag{2.5.7}$$

$$\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = \begin{bmatrix} D_x & 0 & 0 \\ 0 & D_y & 0 \\ 0 & 0 & D_z \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix},$$

where D_b is the quadrotor's drag effect on body frame, and v is velocity of the quadrotor.

In terms of rotational motion of the quadrotor, by using (2.5.2) and from [22,35], we can obtain the body frame angular acceleration equations of the quadrotor as follows:

$$\dot{w} = J^{-1}[M - w \times Jw], \tag{2.5.8}$$

where w is angular velocity, and J is a rotational inertia matrix in the body frame.

Thus, by considering drag effects on the rotational motion and disturbance effect from each rotor angular velocity, we can obtain the rotation equations of the quadrotor as follows:

$$\begin{bmatrix} J_x\ddot{\theta} \\ J_y\ddot{\phi} \\ J_z\ddot{\psi} \end{bmatrix} = \begin{bmatrix} (J_y - J_z)\dot{\phi}\dot{\psi} \\ (J_z - J_x)\dot{\theta}\dot{\psi} \\ (J_x - J_y)\dot{\theta}\dot{\phi} \end{bmatrix} + \begin{bmatrix} l(T_1 - T_2) \\ l(T_3 - T_4) \\ K_\psi(T_1 + T_2 - T_3 - T_4) \end{bmatrix} + \begin{bmatrix} -J_r\dot{\phi}\Omega \\ J_r\dot{\theta}\Omega \\ 0 \end{bmatrix} - \begin{bmatrix} f_\theta \\ f_\phi \\ f_\psi \end{bmatrix}, \tag{2.5.9}$$

where J_r is inertial moment for each rotor, K_ψ is positive gain for yaw motion, Ω is disturbance effect from each rotor and $f = [f_\theta,\ f_\phi,\ f_\psi]^T$ is angular drag force within the rotation equation.

Moreover, these forces are defined from works mentioned earlier [3, 22, 34] as follows:

$$f_a = d_b w, \tag{2.5.10}$$

$$\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = \begin{bmatrix} d_\theta & 0 & 0 \\ 0 & d_\phi & 0 \\ 0 & 0 & d_\psi \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix},$$

where d_b is the quadrotor's rotational drag effect on body frame, and w is angular velocity of the quadrotor.

Afterwards, from (2.5.6) and (2.5.9), we describe non-linear dynamic representations of the overall system as follows:

$$\begin{aligned}
\ddot{x} &= \frac{(T_1+T_2+T_3+T_4)(sin\psi sin\phi + cos\phi sin\theta cos\psi)}{m} - d_x\dot{x}, \\
\ddot{y} &= \frac{(T_1+T_2+T_3+T_4)(-sin\phi cos\psi + cos\phi sin\theta sin\psi)}{m} - d_y\dot{y}, \\
\ddot{z} &= \frac{(T_1+T_2+T_3+T_4)(cos\phi cos\theta)}{m} - g - d_z\dot{z}, \\
\ddot{\theta} &= \frac{(J_y-J_z)\dot{\phi}\dot{\psi}}{J_x} + \frac{l(T_1-T_2)}{J_x} - \frac{J_r\dot{\phi}\Omega}{J_x} - d_\theta\dot{\theta}, \\
\ddot{\phi} &= \frac{(J_z-J_x)\dot{\theta}\dot{\psi}}{J_y} + \frac{l(T_3-T_4)}{J_y} - \frac{J_r\dot{\theta}\Omega}{J_y} - d_\theta\dot{\phi}, \\
\ddot{\psi} &= \frac{(J_x-J_y)\dot{\phi}\dot{\psi}}{J_z} + \frac{K_\psi(T_1+T_2-T_3-T_4)}{J_z} - d_\theta\dot{\theta}.
\end{aligned} \tag{2.5.11}$$

Finally, because of the quadrotor vehicle at low speeds and ignoring inertial, drug and Coriolis effects [18, 22], we can obtain the simplified nonlinear model of the quadrotor system as follows:

$$\begin{aligned}
\ddot{x} &= \frac{(T_1+T_2+T_3+T_4)(sin\psi sin\phi + cos\phi sin\theta cos\psi)}{m}, \\
\ddot{y} &= \frac{(T_1+T_2+T_3+T_4)(-sin\phi cos\psi + cos\phi sin\theta sin\psi)}{m}, \\
\ddot{z} &= \frac{(T_1+T_2+T_3+T_4)(cos\phi cos\theta)}{m} - g, \\
\ddot{\theta} &= \frac{l(T_1-T_2)}{J_x}, \\
\ddot{\phi} &= \frac{l(T_3-T_4)}{J_y}, \\
\ddot{\psi} &= \frac{K_\psi(T_1+T_2-T_3-T_4)}{J_z}.
\end{aligned} \tag{2.5.12}$$

2.5.2 Linearization of The Quadrotor Model

In the experimental studies, in consideration of the simplified nonlinear model of system (2.5.12), it is assumed that rotational angles are close to zero since small angle approximation is used to linearize quadrotor vehicles [18, 22]. Therefore, we obtain the linearized model of the Qball-X4 quadrotor vehicle system. Then, we easily adapt the linearized model into each vehicle controller for performing attitude and altitude behaviors of the Qball-X4 quadrotor vehicle in real-time implementations.

Actuators: Hovering the Qball-X4 quadrotor vehicles, we use mounted motors and propellers which works clockwise and counter-clockwise in pairs. In order to generate thrust forces by using actuators, the first-order system represents the thrust models [25] as follows:

$$T_i = K\frac{w}{s+w}u_i,\ i = 1, .., 4, \tag{2.5.13}$$

where u_i is the pulse-width modulation (PWM) input for each actuator, w is the actuator bandwidth and K is a positive gain. In addition, in order to use this for pitch and roll control design, we can use a state variable v, which represents actuator dynamics as follows:

$$v = \frac{w}{s+w}u. \tag{2.5.14}$$

With laplace form being changed to dynamic state-space form, input dynamic is written as follows:

$$\dot{v} = -wv + wu. \tag{2.5.15}$$

Furthermore, from altitude and attitude control schemes which will be designed in the following chapters, by using controlled inputs $u_\theta,\ u_\phi,\ u_\psi\ and\ u_z$, we can combine these control inputs to each motor input for generating corresponding thrust forces in a real system as follows:

$$\begin{aligned} u_1 &= u_\theta + u_\psi + u_z, \qquad (2.5.16)\\ u_2 &= -u_\theta + u_\psi + u_z,\\ u_3 &= u_\phi - u_\psi + u_z,\\ u_4 &= -u_\phi - u_\psi + u_z. \end{aligned}$$

Thus, these obtained motor inputs by help of actuator dynamic (2.5.13) command the quadrotor vehicles' attitude and altitude control schemes.

Pitch and Roll Models: Driving the Qball-X4 quadrotor system from forward to backward or left to right direction, roll and pitch models provide motion of the quadrotor in x-y axes. When considering the roll and pitch rotations, it is assumed that these models are decoupled to examine motion of the quadrotor separately in the x-y plane as demonstrated in Figure 2.9. Hence, the rotation is generated around the center of gravity by the differences of thrust force pairs. Furthermore, from (2.5.12), we obtain linearized equations of pitch/roll motion dynamics because of the quadrotor at low speeds and by neglecting other dynamic effects as follows:

$$J_\zeta \ddot{\zeta} = L \,\triangle T_\zeta, \qquad (2.5.17)$$

where ζ represents θ and ϕ rotational angles of pitch/roll motion, J_ζ represents $J_\theta = J_x$ and $J_\phi = J_y$ which are the rotational inertia of the quadrotor in pitch/roll motion, and L is the distance between the center of gravity and thrust force pairs. Also, $\triangle T_\zeta$ is the difference between responsible force pairs for pitch/roll motion. Therefore, from (2.5.13), we write this as follows:

$$\triangle T_\zeta = K(\frac{s}{s+w}\triangle u_\zeta), \qquad (2.5.18)$$

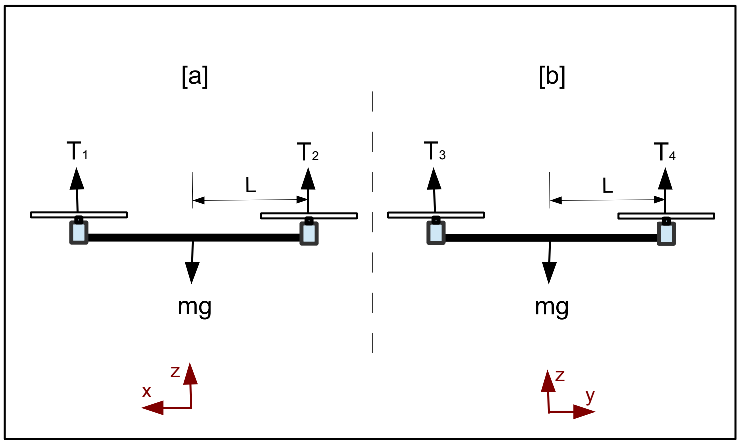


Figure 2.9: Illustration [a] pitch and [b] roll motions

where $\triangle T_\theta = (T_1 - T_2)$ and $\triangle T_\phi = (T_3 - T_4)$ are thrust force differences for pitch/roll motion, and from (2.5.16), $\triangle u_\zeta$ represents inputs from the roll/pitch model where $\triangle u_\theta = (u_1 - u_2) = 2(u_\theta)$ and $\triangle u_\phi = (u_3 - u_4) = 2(u_\phi)$.

By combining dynamics from (2.5.17) and (2.5.18), and adding integrator feedback structure for disturbance effect cancellation, it is written as

$$\dot{s} = \zeta, \qquad (2.5.19)$$

finally, we obtain the system dynamics in the generic state-space of pitch/roll motion in order to use in a linear controller design as follows:

$$\dot{x}_\zeta = A_\zeta x_\zeta + B_\zeta u_\zeta, \qquad (2.5.20)$$

$$x_\zeta = \begin{bmatrix} \zeta \\ \dot{\zeta} \\ v \\ s \end{bmatrix}, \quad A_\zeta = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{2KL}{J_\zeta} & 0 \\ 0 & 0 & -w & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad B_\zeta = \begin{bmatrix} 0 \\ 0 \\ w \\ 0 \end{bmatrix}. \tag{2.5.21}$$

where ζ represents θ and ϕ rotational angles of pitch/roll motion, J_ζ represents J_θ and J_ϕ which are the rotational inertia of the quadrotor in pitch/roll motion, u_ζ represents u_θ and u_ϕ which are inputs of the quadrotor in pitch/roll motion, and L is the distance between the center of gravity and thrust force pairs.

Yaw design: In order to control of the quadrotor in yaw motion, we use a relationship between PMW inputs and torques [25] as follows:

$$\tau_i = K_\psi T_i = K_\psi K \frac{w}{s+w} u_i, \; i = 1, .., 4. \tag{2.5.22}$$

where τ is generated torques for each actuator, u is control input for each actuator , and K_ψ is positive gain. After torques $\tau_i, \quad i = 1, ..., 4$ are generated by each motor, according to quadrotor design, one pair of motor works clockwise and the other pair works counter-clockwise. Thus, a difference exists between motor pairs, and this difference causes yaw motion of the quadrotor as seen in Figure 2.10. Moreover, from (2.5.12), we write linearized form of yaw motion as follows:

$$J_\psi \ddot{\psi} = \triangle\tau, \tag{2.5.23}$$

where ψ is the rotational angle in yaw motion, $J_\psi = J_z$ is the rotational inertia of the quadrotor in yaw motion. Also, $\triangle\tau$ is the difference between torque pairs. Therefore, from (2.5.22), we write the torque difference for yaw dynamics as

$$\triangle\tau = K_\psi(T_1 + T_2 - T_3 - T_4) = K_\psi K(\frac{w}{s+w} \triangle u_\psi), \tag{2.5.24}$$

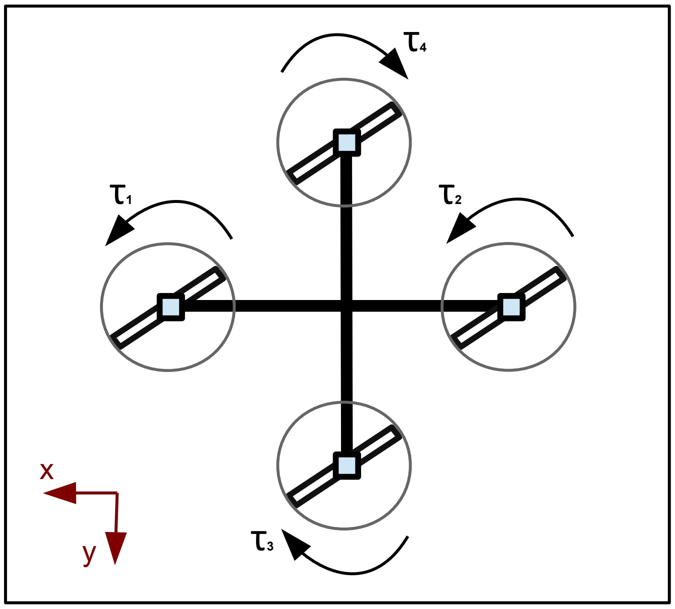


Figure 2.10: Actuator torques for the yaw motion model

where $\triangle u_\psi = (u_1 + u_2 - u_3 - u_4) = 4(u_\psi)$ represents yaw model input.

Then, by using (2.5.23) and (2.5.24), we can write linearized yaw model in form of a transfer function as

$$\psi = \frac{4K_\psi K w}{s^2(s+w)J_\psi} u_\psi. \tag{2.5.25}$$

Altitude Model: Moving the quadrotor vehicle in z-axes, we use the linearized form of height model which is obtained from the simplified nonlinear model from (2.5.12). Furthermore, all actuator thrusts affect height motion in positive direction as seen in Figure 2.9. From (2.5.12), we write the nonlinear form of height model as

$$m\ddot{z} = \triangle T_z cos\theta cos\phi - mg, \tag{2.5.26}$$

where T_z is the total thrust generated by each motor, θ and ϕ represent pitch/roll angles, m is the mass of the quadrotor, and z is the height of the quadrotor.

Assuming that we use small angle approximation and taking the effect of gravity as an offset in linearized form of height model, from (2.5.12) we obtain the simplified linear form of the height model as follows:

$$m\ddot{z} = \triangle T_z = (T_1 + T_2 + T_3 + T_4) = K(\frac{w}{s+w}\triangle u_z), \tag{2.5.27}$$

where $\triangle u_z = (u_1 + u_2 + u_3 + u_4) = 4(u_z)$ represents the height model input.

Then, by using (2.5.27), we write the linearized yaw model in the form of a transfer function as

$$z = \frac{4Kw}{s^2(s+w)m}u_z. \tag{2.5.28}$$

Chapter 3

Real-Time Rigid Formation Control Design

In a rigid and persistent formation control structure in real-time application, control law and its performance directly depend on the agents' communication or sensing structure in the experimental environment. Therefore, in order to perform our formation control structure of the multi-quadrotor system, in this chapter we focus on autonomous formation control of multi-quadrotor systems in real-time implementation and its realistic simulation, and then we develop a distributed formation control law based on leader-follower formation structure.

In formation control problems, in attempt to satisfy the desired shape based on formation control structure, agents have to maintain minimal requirements of desired distances between each responsible pair as discussed in Chapter 2. Therefore, in this chapter, we first provide a formation structure and model for our multi-quadrotor swarm. As mentioned in Chapter 2, there are two types of distance control: symmetric and asymmetric.

In this work, we use the asymmetric control structure as seen in Figure3.1 for the sake of decreasing the control and communication complexities during the experiment. Then, we define our formation problem making some assumptions as the requirements of the testbed. Afterwards, in terms of formation control problems, we have two main parts in the overall control architecture to achieve the purpose of the formation mission, namely, high-level and low-level control schemes as seen in the generic block diagram Figure 3.2. By using testbed facilities as mentioned in Chapter 2, at first, we design the high-level controller responsible to generate desired trajectories for each of the followers by assisting the asymmetric decentralized control architecture. Then, at the low-level control, after reference angles are generated by using desired positions, particular controllers are designed for controlling attitude and altitude models of quadrotor.

As stated earlier, after the overall control design, we are able to perform the autonomous formation control performances of the multi-quadrotor system in real-time implementation, and its realistic simulation.

3.1 Graph Modeling of the Multi-Quadrotor System

In this study, in order to make a formation structure in multi-quadrotor system for simulation and experiment, we consider a swarm S that comprises up to 3 quadrotors each of which is denoted as $A_i,\ i = 1, ..., 3$ as seen in Figure 3.1 where each quadrotor agent moves at a constant height in $\mathbb{R}^3$. Moreover, we denote the inter-agent distances as d_{ij}, $\overrightarrow{(ij)} \in E_F$; $i, j = 1, ..., 3$, and the position vector of each quadrotor as $P_i(t) = [x_i(t)\ y_i(t)\ z_i]^T$, $i = 1, ..., 3$ in the quadrotor swarm. As mentioned in Chapter 2, concerning formation graph, it is assumed that the swarm system has a directed graph $G_F = (V_F, E_F)$, where each vertex represents agents A_i, $i \in V_F$; $i = 1, ..., 3$ and each edge represents directed inter-agent distances d_{ij}, $\overrightarrow{(ij)} \in E_F$; $i,j = 1,...,3$ as seen in Figure 3.1. Therefore, we represent the formation $F = (S, G_F, D_F)$ as the combination of swarm S, the underlying directed graph $G_F = (V_F, E_F)$ and desired distance set $D_F = \{d_{ij} | \overrightarrow{(ij)} \in E_F\}$.

Note that the multi-quadrotor swarm system has an asymmetric formation control architecture where one of corresponding agent pair is responsible for maintaining the inter-agent distance in keeping formation shape as in [1]. In other words, the responsible follower is supposed to maintain inter-agent distance during the motion for each time instant.

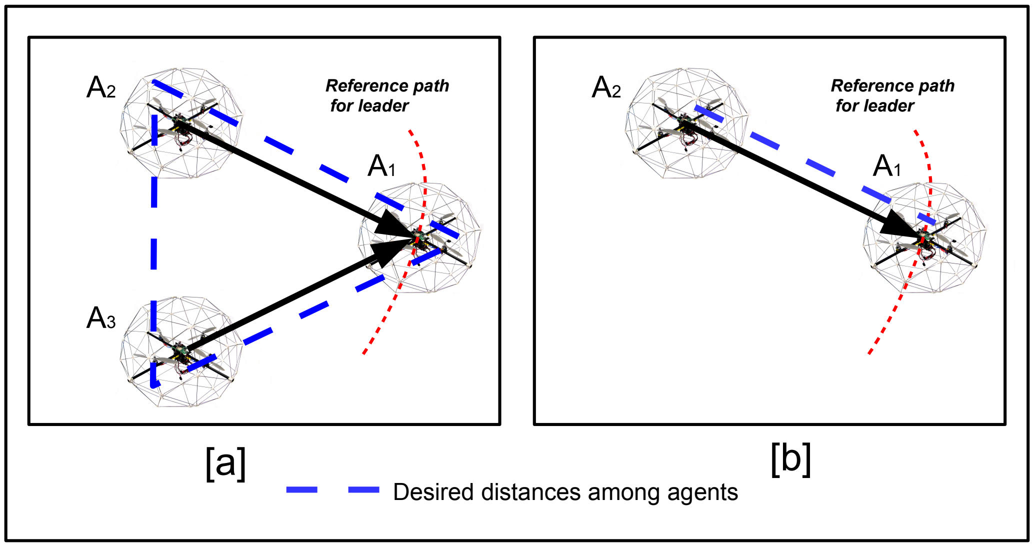

A2
Reference path for leader
A1
A3
A2
Reference path for leader
A1
[a]
[b]
Desired distances among agents

Figure 3.1: Directed underlying graphs for [a] equilateral triangular formation control of a 3-quadrotor system and [b] distance keeping for a 2-quadrotor system

3.2 Formation Control Problem

The main goal of this study is that the leader tracks the desired way-points, and the follower quadrotors maintain the predefined formation shape in real-time implementation and simulation under the given assumptions. In three-agent or two-agent quadrotor systems, each agent is described as A_i, $i = 1, 2, 3$, and their position information are denoted P_i=($x_i(t)$,$y_i(t)$ z_i), $i = 1, 2, 3$ at any instant of time. During this work, we denote the inter-agent distances d_{ij}, $i = 1, 2, 3,\ j = i = 1, 2, 3$ as seen in Figure 3.1 depending on the leader-follower formation structure which is explained in Chapter 2. in our testbed, by the help of the indoor localization facilities as mentioned in Chapter 2, each quadrotor knows its own global position as well as each neighbor quadrotor as seen in (3.2). Also, because of omni-directional motion ability of the quadrotor system, relative position information will be enough in each neighbor agent pair for keeping rigid and persistent formation structure. Therefore, considering the control task, we can use relative position information among the agents to achieve the high-level control task. In that case, relative positions between responsible quadrotor pairs are represented as r_{ij}, $i = j = 1, 2, 3$, where $r_{ij} = -r_{ji}$ as $r_{12}(t) = (x_1(t) - x_2(t), y_1(t) - y_2(t))$. The formation control law aims to ensure that

$$\lim_{t \longrightarrow \infty} |\|r_{ij}\| - \|r_{dij}\|| = 0, \tag{3.2.1}$$

where r_{ij} is the relative position, and r_{dij} is the desired relative position between a responsible quadrotor pair.

Due to the localization and formation problem constraints, for formation studies, we have some assumptions as follows:

Assumption 3.2.1 *It is assumed that quadrotors establish formation roughly before motion. In other words, the quadrotor swarm starts close to initial positions at the edges of* *the desired distance and triangular formation shape for real-time implementation and its simulation.*

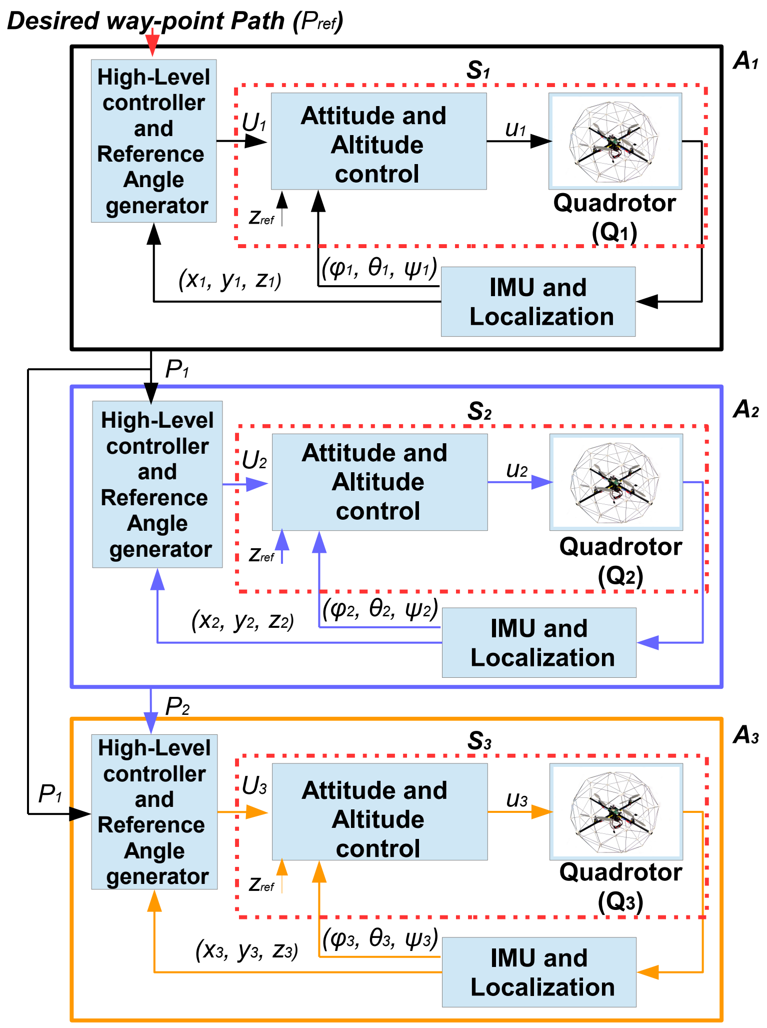
Assumption 3.2.2 *In the formation shape, each quadrotor A_i, $i = 1, 2, 3$ has its own global position information as $P_i(t) = [x_i(t)\ y_i(t)\ z_i]^T$ at any time t, and assumed that all quadrotors move a constant height z during the motion. Therefore, the formation task performance is in the $x-y$ plane, $\mathbb{R}^2$. Furthermore, the agents' desired trajectory generation is derived considering constant height z_i, and motion in $P_i = (x_i, y_i)$.*

Assumption 3.2.3 *We also assume that each quadrotor A_i, $i = 1, 2, 3$ in the formation shape knows its neighbor's position information by using a testbed and indoor localization system at any time, t.*

Considering all assumptions above, we can now summarize the formation control problem.

Problem 3.2.1 *[Formation Maintenance Problem] Consider three-quadrotor swarm system S consists of A_1, A_2, and A_3 within formation structure as seen in Figure 3.1, and the overall controller architecture is as seen in Figure 3.2. Let $F = (S, G_F, D_F)$ is a rigid and persistent formation in the leader-follower structure, where D_F is consistent inter-agent distance. Under the assumptions 1, 2, and 3, generate the online desired position and then reference angles with respect to D_F for each follower quadrotor. Furthermore, with respect to generated desired angles and individual altitude and attitude controller, generate controlling PWM inputs as in* (2.5.16) *so that the swarm S satisfies condition in* (3.2.1) *during the formation task.*

Therefore, the formation maintenance problem has two parts; (*i*) at the high-level, the controller is responsible to generate the desired position and angles during the motion, (*ii*) at the low-level, the controller is responsible for tracking the performance of quadrotors for the maintenance of distance and equilateral triangle formation by using individual controllers.



Desired way-point Path (P_ref)
A1
S1
High-Level controller and Reference Angle generator
U1
Attitude and Altitude control
u1
Quadrotor (Q1)
z_ref
(φ1, θ1, ψ1)
(x1, y1, z1)
IMU and Localization
P1
A2
S2
High-Level controller and Reference Angle generator
U2
Attitude and Altitude control
u2
Quadrotor (Q2)
z_ref
(φ2, θ2, ψ2)
(x2, y2, z2)
IMU and Localization
P2
P1
A3
S3
High-Level controller and Reference Angle generator
U3
Attitude and Altitude control
u3
Quadrotor (Q3)
z_ref
(φ3, θ3, ψ3)
(x3, y3, z3)
IMU and Localization

Figure 3.2: Block diagram of the control system

3.3 On-line Trajectory Generation for the Distributed Control Scheme

This section is about the multi-quadrotor swarm system's high-level control design. It is considered that the high-level controller is a decentralized, hierarchical, asymmetric control structure to solve *Problem* (3.2.1). Therefore, we designate the agents as Leader A_1, first follower A_2 and second follower A_3 respectively. Under consideration of formation *Problem* (3.2.1), it is necessary to maintain the inter-agent distances for each agent $A_i \;\; i = 1, 2, 3$ throughout the autonomous motion. Hence, by using the host computer communication, localization system, decentralized control architecture and relative position knowledge, we can derive desired global positions for the members of swarm as follows:

Leader Quadrotor's Control Law: At first, leader quadrotor A_1 is responsible to satisfy reference path tracking, so the leader should satisfy

$$\|P_1 - P_{ref}\| \leqslant \overline{P}, \tag{3.3.1}$$

where P_1 is the leader's global position, P_{ref} is a predefined way-point for the leader's motion, and $\overline{P}$ is a small threshold value for tracking.

In this respect, the leader agent's tracking can be derived at any time instant by comparing the predefined way-point path P_{ref} and threshold value $\overline{P}$ as

$$\begin{aligned} if \;\; \|P_1(t) - P_{ref}(t)\| \geqslant \overline{P}, &\;\; P_{1d}(t) = P_{ref}(t) \\ otherwise, &\;\; P_{1d}(t) = P_{ref}(t+1). \end{aligned} \tag{3.3.2}$$

where $P_{1d}(t)$ is the generated desired position.

Follower Quadrotors' Control Law: The other quadrotors, $A_i, \;\; i = 2, 3$ as ordinary agents follow the leader in a hierarchy as per the formation distances D_F. Therefore, followers $A_i, \quad i = 2,3$ are required to move on desired positions $P_{id}, \quad i = 2,3$, and then they can be satisfied condition (3.2.1) during the formation task. As mentioned in Section 3.2, using global position informations and desired relative positions will be enough to generate the desired positions of the follower agents, and then this will satisfy rigid and persistent formation maintenance. We obtain the desired positions of the follower agents P_{id} at any time instant as,

$$\begin{aligned} P_{2d}(t) &= P_1(t) + r_{d12}, \\ P_{3d}(t) &= P_1(t) + r_{d13}, \end{aligned} \tag{3.3.3}$$

where $r_{d1i}, \ i = 2,3$ is the desired relative position between the leader and followers.

3.4 Reference Angle Generation

By distributing the high-level controller in Section 3.3, we obtain the desired global positions for each quadrotor agent. In this section, by using these generated desired positions from (3.3.2) and (3.3.3), and actual positions from localization system, we can obtain the desired angles $U_i = (\theta_{id}, \ \phi_{id}, \ \psi_{id}), \quad i = 1,2,3$ as seen in block diagram Figure 3.2 before individual attitude controllers. Hence, in order to generate the reference angles of each quadrotor, we consider the PD controller as follows:

$$\begin{aligned} \phi_{di} &= k_{pi}(y_{di} - y_i) + k_{di}(\dot{y_{di}} - \dot{y_i}), \qquad (3.4.1) \\ \theta_{di} &= k_{pi}(x_{di} - x_i) + k_{di}(\dot{x_{di}} - \dot{x_i}), \\ \psi_{di} &= 0. \end{aligned}$$

Yaw motion does not directly affect the quadrotor motion on $x - y$ axes. For this reason, it is assumed that desired yaw angle is $\psi_{di} = 0, \ i = 1,2,3$ for $\forall t > 0$.

3.5 Distributed Control Scheme for Attitude and Altitude Control Design

In this section, considering *Problem* (3.2.1), we now design of the attitude and altitude control laws for formation performance of each quadrotor vehicle after the high-level control as seen in Figure 3.2.

Pitch and Roll Control: Considering the linearized dynamic equation (2.5.17), we first present a linear quadratic controller (LQR) design to generate effective control inputs of the pitch and roll model, which are u_θ and u_ϕ. In order to design a linear quadratic control scheme, the state-space form (2.5.21) of linearized dynamic representation are shown in the generic state-space form as

$$\dot{x}_\zeta = A_\zeta x_\zeta + B_\zeta u_\zeta, \tag{3.5.1}$$

where A_ζ is a 4x4 matrix and B_ζ is a 4x1 matrix, and for minimizing performance measurement of the system, we can use the cost function as

$$J = \int (x^T Q x + \lambda u^2) dt, \tag{3.5.2}$$

where Q is a 4x4 matrix positive semidefinite matrix and λ is a positive scalar.

Then, in order to find stable feedback gain, a state-feedback control law is written as,

$$u = -K_c x, \tag{3.5.3}$$

where K_c is an 1x4 gain matrix obtained by

$$K_c = \lambda^{-1} B_\zeta^T P. \tag{3.5.4}$$

Here, P is a 4x4 auxiliary matrix obtained by solving the Riccati equation

$$A_\zeta^T P + PA_\zeta - PB_\zeta \lambda^{-1} B_\zeta^T P + I = 0, \quad P = P^T > 0. \tag{3.5.5}$$

Yaw Control: As a result of not directly affecting the x-y motion of the quadrotor, we consider that the yaw motion is more free. Therefore, the yaw controller is taken as proportional (P) control law by using (2.5.25) as

$$u_\psi = K_p(\psi_{di} - \psi_i). \tag{3.5.6}$$

Altitude Control: By assuming 3.2.2, altitude controller is derived for keeping quadrotors in constant height during the task. Therefore, we consider that the system's altitude controller is proportional-integral-derivative (PID) control law by using (2.5.28) as

$$u_z = K_p(z_{di} - z_i) + K_i \int_0^t (z_{di} - z_i)dt + K_d(\dot{z_{di}} - \dot{z}_i). \tag{3.5.7}$$

After generating of control inputs $u_\theta,\ u_\phi,\ u_\psi and\ u_z$, by using the altitude and attitude control scheme as above, we combine these inputs from (2.5.16) to generate each motor PWM inputs for the Qball-X4 quadrotor system. Therefore, these motor inputs obtained by help of actuator dynamic (2.5.13) command the system's attitude and altitude behavior during the motion of the quadrotor system.

3.6 Simulation and Experimental Studies

In this section, we present the results of formation performances of multi-quadrotor swarm system for different reference path in real-time implementation and its simulation. Before the experiment, we employ the formation task for the Qball-X4 quadrotor dynamic model described in Chapter 2 by using the MATLAB *Simulink*® simulation environment. Making a more realistic simulation study, we add characterized Gaussian noises into the system's pitch and roll dynamics, and these system noises come from the IMU measurement in real-time applications. Thus, determining the noise condition, we use one of the Qball-X4 quadrotor's roll angle data for the preliminary 25 [sec] as in Figure 3.3 which is measured when the system does not hover at first. After receiving the data, we calculate the variance and mean value as below.

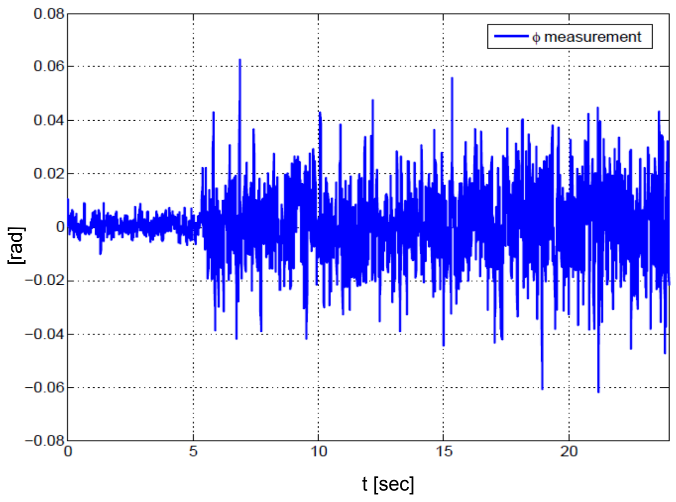


Figure 3.3: Qball-X4 roll angle measurement for 25[sec]

$$\mu = \frac{1}{n}\sum_{i=1}^{n}(x_i) \tag{3.6.1}$$

$$Var(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{3.6.2}$$

where $Var(X)$ is the variance value of the measurement data and μ is the average of the noisy measurement data.

By using measured roll data, and from (3.6.1) and (3.6.2), we can find IMU's noise characteristics which are calculated as $\mu = 0.000721$ and $Var(X) = 0.0003$. Afterwards, we add these characteristics into the quadrotors' pitch and roll dynamics in simulation by the Gaussian noise model.

At first, we employ two different paths for the leader quadrotor, namely, spiral and line path in simulation model. We also designed the high-level and low-level controllers from sections 4.3, 4.4, and 4.5 in order to maintain the desired equilateral triangle with each edge being 3 meter during the motion. Second, in the experiment, we have just employed a line path because of the localization constraint for leader quadrotor. We are able to use two quadrotors for the same reason. In addition, we use the same designed controller structures to maintain rigid formation distance 2 meter between the leader and follower.

Moreover, considering the QBall-X4 quadrotor, in order to design experiment and simulation, we use system dynamic parameters which are specified by Quanser Inc. as seen in table (3.1).

Table 3.1: System dynamic parameters [25]

K	ω	J_{roll}	J_{pitch}	M	K_y	J_{yaw}	L
120 N	15 rad/sec	0.03 kgm^2	0.03 kgm^2	1.4 kg	4 Nm	0.04kgm^2	0.2 m

Furthermore, before the controller design, we use angle generation with $k_p = 0.7$ and $k_d = 0.4$. Then, for the LQR controller design of pitch and roll dynamics, the ideal cost is obtained by using randomly found $Q = diag(150\ \ 0\ \ 20000\ \ 25)$ and $\lambda = 30000$. Afterwards, by (4.3.5), LQR gains are calculated as $K_c = diag(0.0763\ \ 0.0143\ \ 1.1726\ \ 0.0288)$, and all poles are placed to the left-hand-side of the complex palne. Also, the yaw proportional control gain is taken as $K_{p\psi} = 0.015$, and for the altitude PID controller, $K_{pz} = 0.006$, $K_{iz} = 0.008$,and $K_{dz} = 0.002$.

From the line path results of the next Section 3.6.1 and 3.6.2, by using mean square error calculation (3.6.3) and the formation distance errors of experiment and simulation, we obtain the formation performance of the multi-quadrotor swarm system. Thus, for simulation and experiment, formation mean square errors are demonstrated in table (3.2).

$$mse = \frac{1}{N}\sum_{i=1}^{N}(\varepsilon_i)^2, \quad \varepsilon_i = \sqrt{(x_i - x_{id})^2 + (y_i - y_{id})^2}, \quad i = 1, ..., 3. \tag{3.6.3}$$

where $\varepsilon_i,\ i = 1, ..., 3$ is the formation distance error between the leader and followers during formation motion, N is the total data number and mse is the mean square error.

Table 3.2: Formation mean squared error performances by the LQR control scheme

Mean squared error	Simulation			Experimental	
	Leader	Follower 1	Follower 2	Leader	Follower 1
Overall term	2.034	0.0072	0.0262	2.12	0.026
After the settling	0.467	0.0038	0.0039	2.04	0.025

3.6.1 Simulation Results

Formation control simulation results are given in Figure 3.4 - 3.11 for spiral and line path. As they are shown, the two-level distributed control structure is able to tackle the formation problem described in Section 3.2. For testing our algorithm, as can be seen in Figure 3.4 and 3.8, our algorithm works satisfactorily under applied Gaussian sensor noise in simulations.

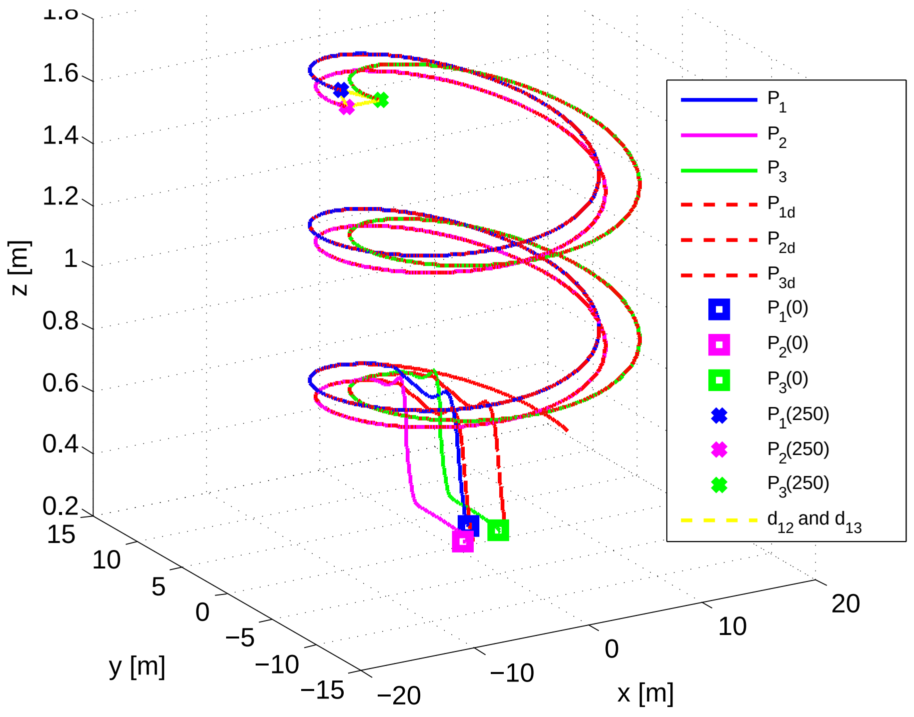
As seen in Figures 3.5 and 3.9, all agents track their references. Also, each follower maintains its desired distance and the desired formation shape with small error as seen in

3.4b and 3.8c. Note that the orientation of formation does not change during the motion, and agents are in trends of tracking of attitude angles as seen in Figures 3.6 and 3.10. Also, note that all control inputs are suitable for real-time implementations.

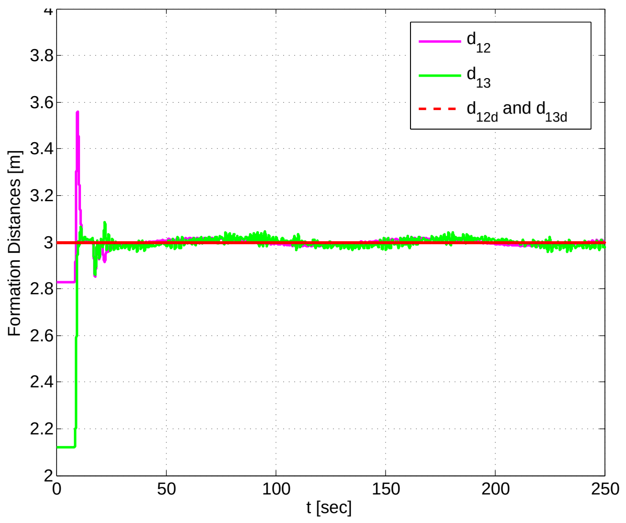
3.6.2 Experimental Results

Formation control experimental results are given in Figure 3.12 - 3.16 for line path. As they are shown, the two-level distributed control structure is able to tackle the formation problem described in Section 3.2. For testing our algorithm, as seen in Figure 3.13, our algorithm works satisfactorily for maintaining the desired rigid distance in real-time environment.

Both of agents track insistently their references as seen in Figure 3.12 after the system ready for hovering at 25 [sec]. Follower maintains its desired distance with small error as seen in 3.13. Note that the orientation of formation does not change during the motion, and agents are in trends of tracking of attitude angles as seen in Figures 3.15.

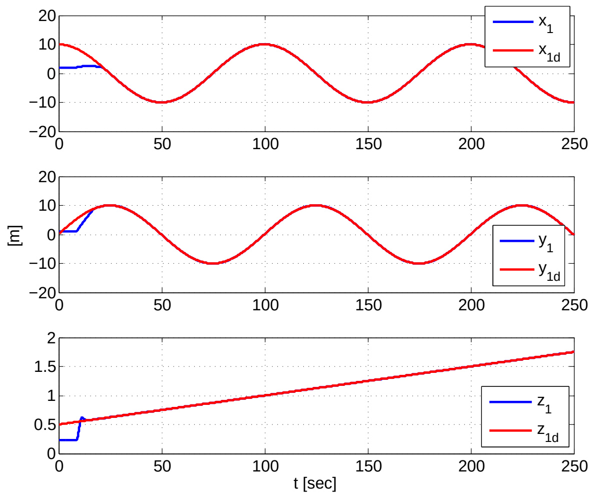


(a) Pathways of agents

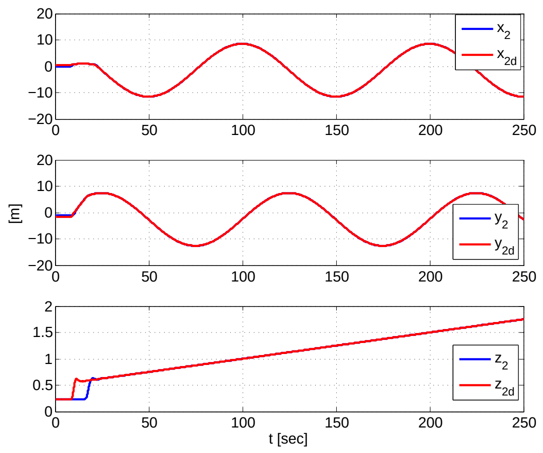


(b) Formation distances among leader and followers

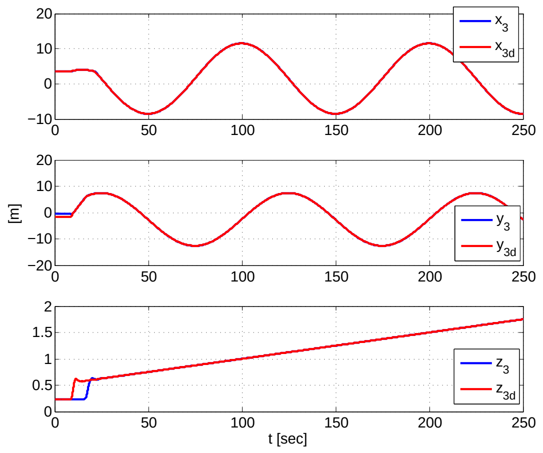
Figure 3.4: Agents' motions and formation distances for spiral path with noise effect



(a) Leader reference tracking

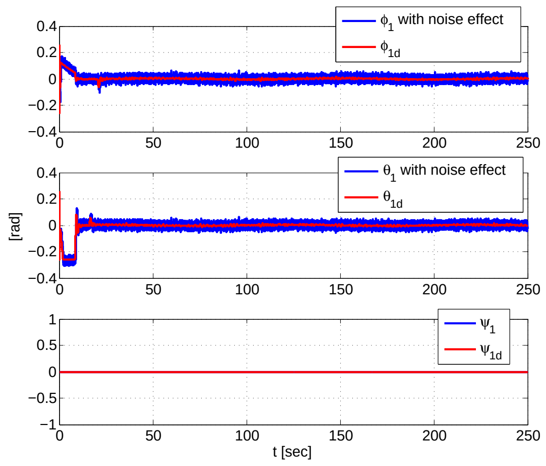


(b) Follower 1 reference tracking

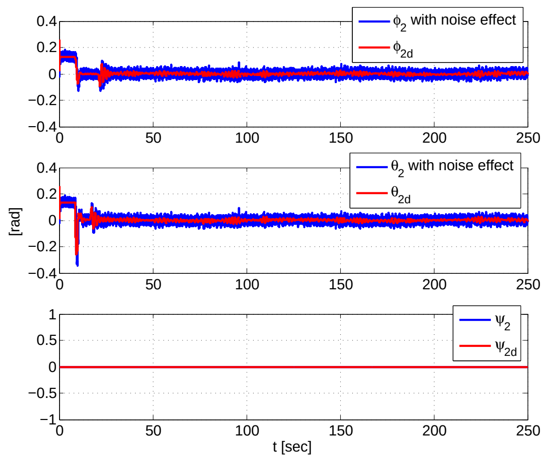


(c) Follower 2 reference tracking

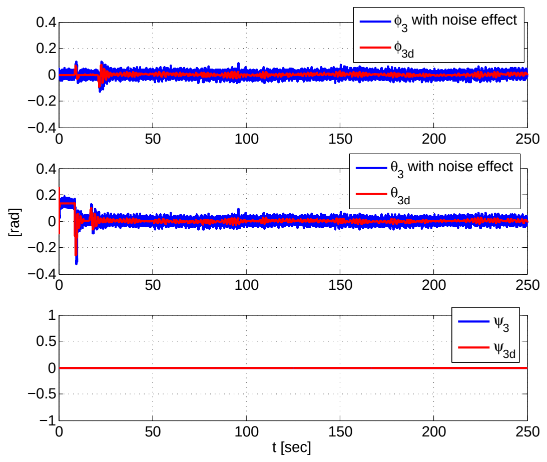
Figure 3.5: Tracking performances of agents for spiral path with noise effect



(a) Leader attitude angles

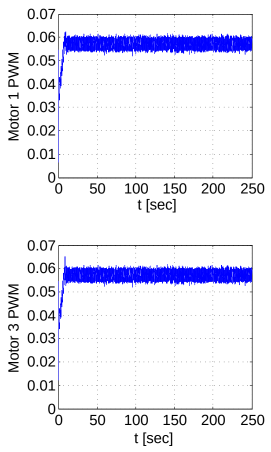


(b) Follower 1 attitude angles

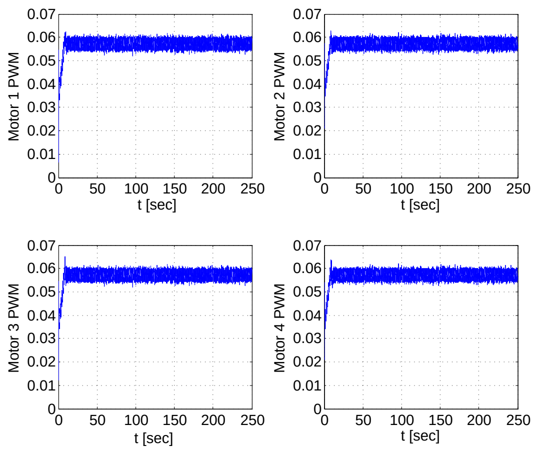


(c) Follower 2 attitude angles

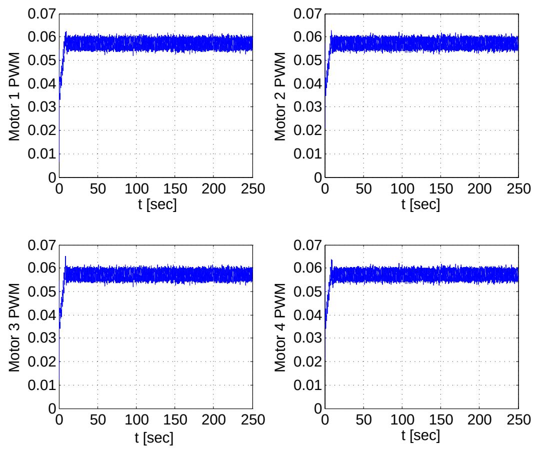
Figure 3.6: Attitude performances of agents for line path with noise effect.



(a) Leader PWM inputs

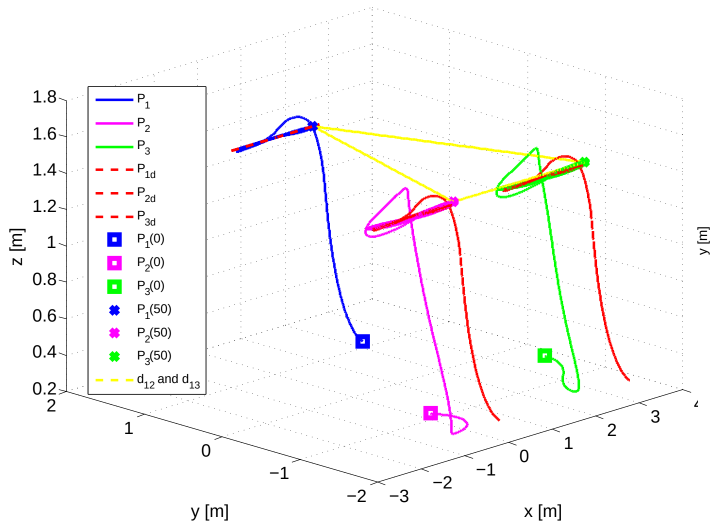


(b) Follower 1 PWM inputs

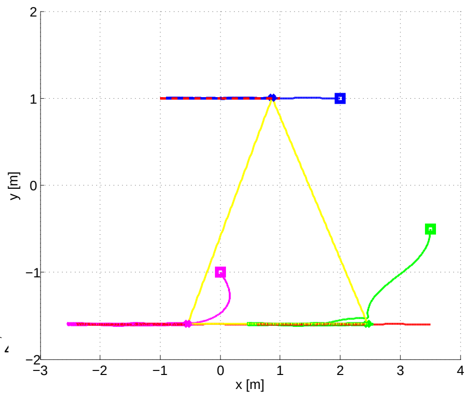


(c) Follower 2 PWM inputs

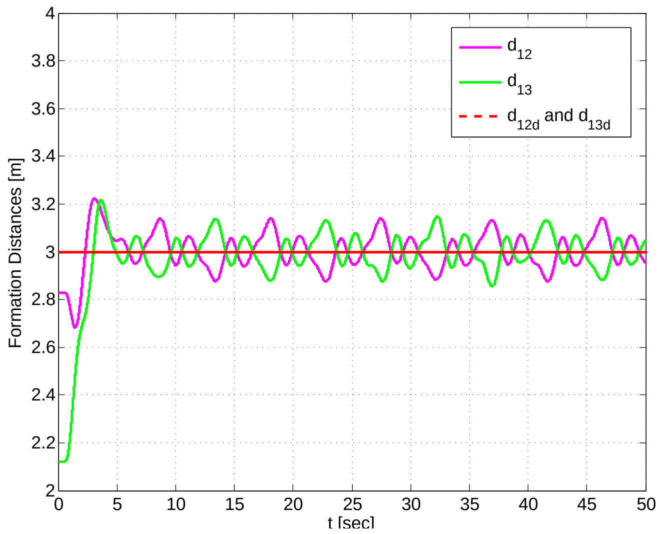
Figure 3.7: Control inputs of agents for line path with noise effect.



(a) Pathways of agents

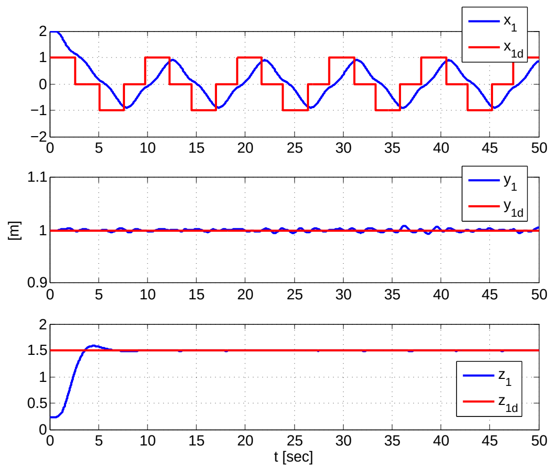


(b) Pathways in x-y motion

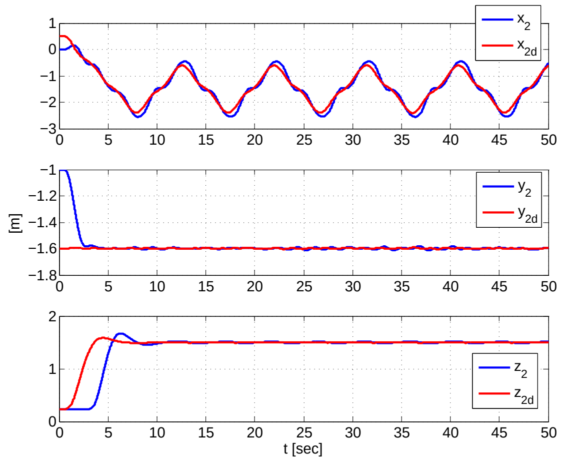


(c) Formation Distance among leader and followers

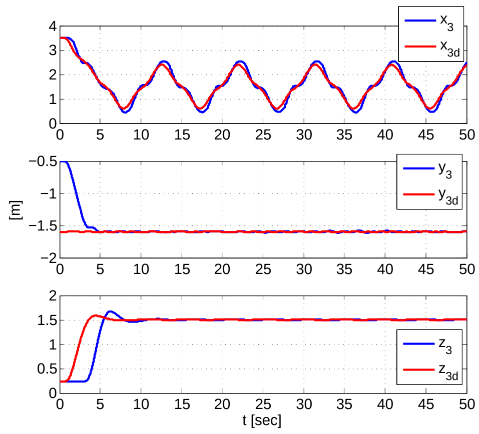
Figure 3.8: Agents' motions and formation distances for line path with noise effect



(a) Leader reference tracking

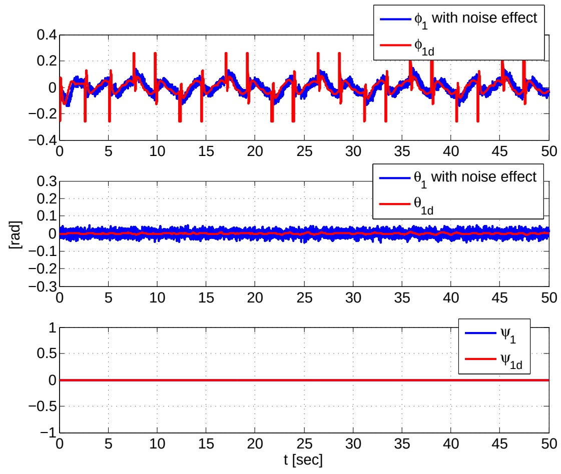


(b) Follower 1 reference tracking

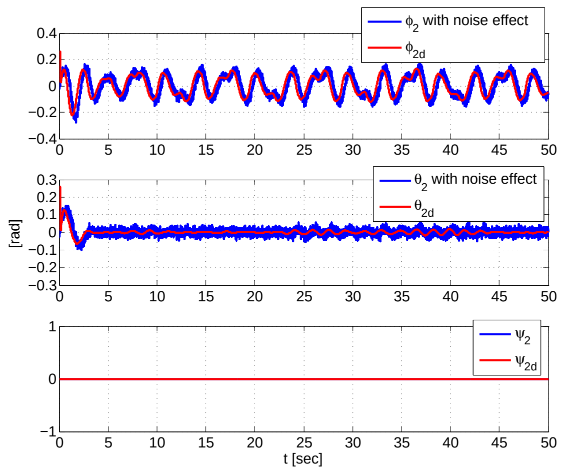


(c) Follower 2 reference tracking

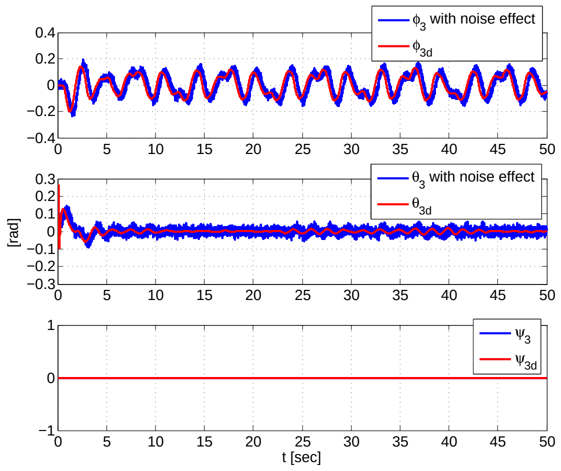
Figure 3.9: Tracking performances of agents for line path with noise effect



(a) Leader attitude angles

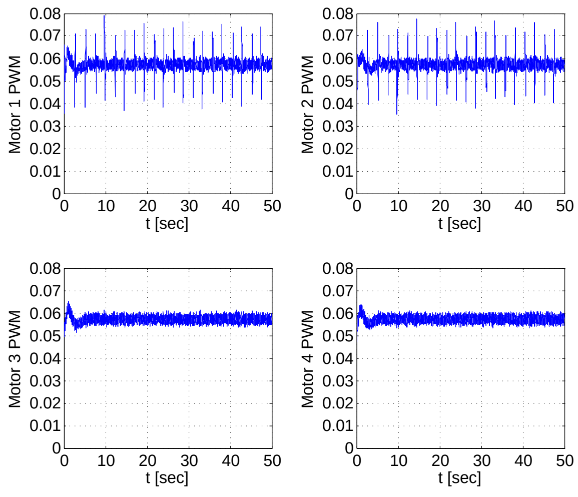


(b) Follower 1 attitude angles

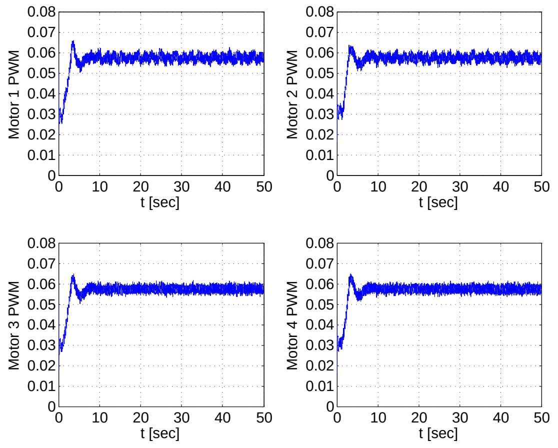


(c) Follower 2 attitude angles

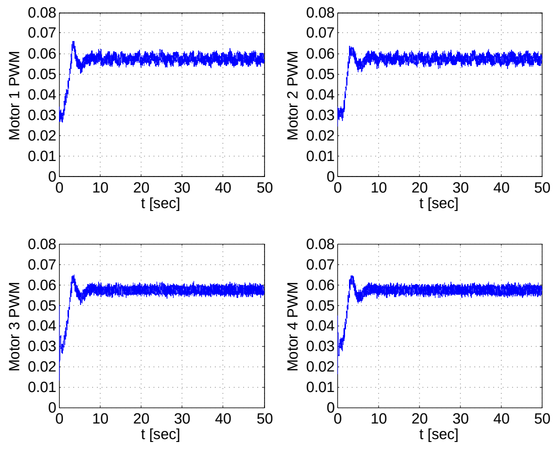
Figure 3.10: Attitude performances of agents for line path with noise effect.



(a) Leader PWM inputs

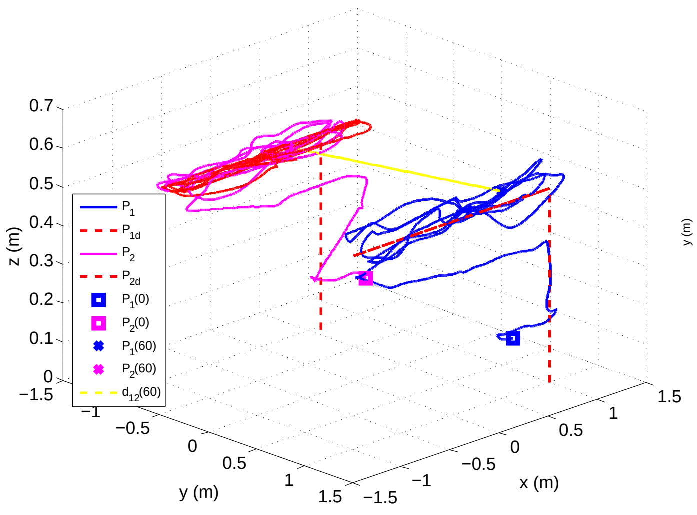


(b) Follower 1 PWM inputs

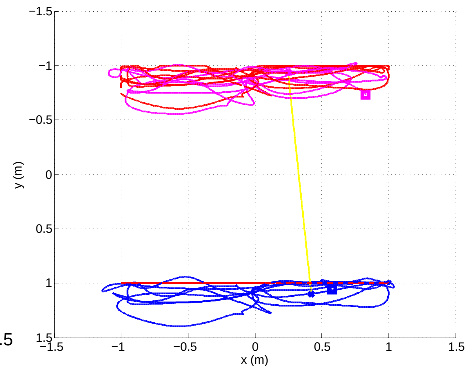


(c) Follower 2 PWM inputs

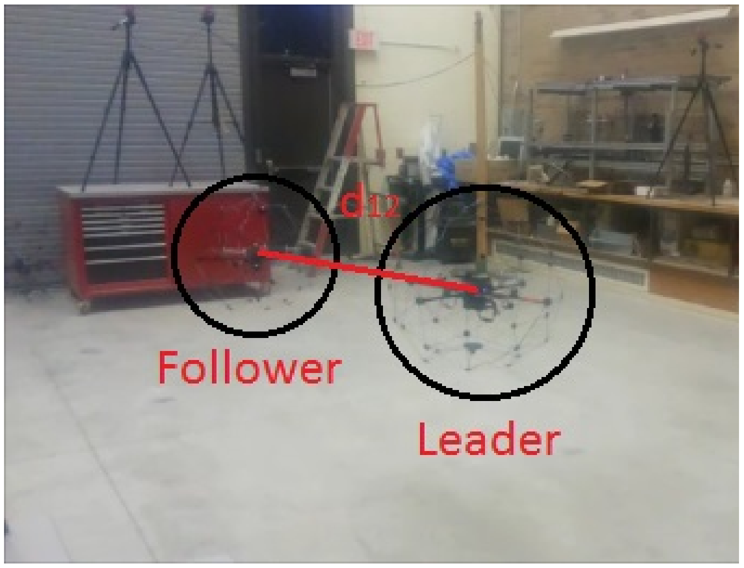
Figure 3.11: Control inputs of agents for line path with noise effect.



(a) Pathways of agents



(b) Pathways in x-y motion



(c) Experimental view of leader and follower during the task

Figure 3.12: Agents' motions for line path in experimental implementation

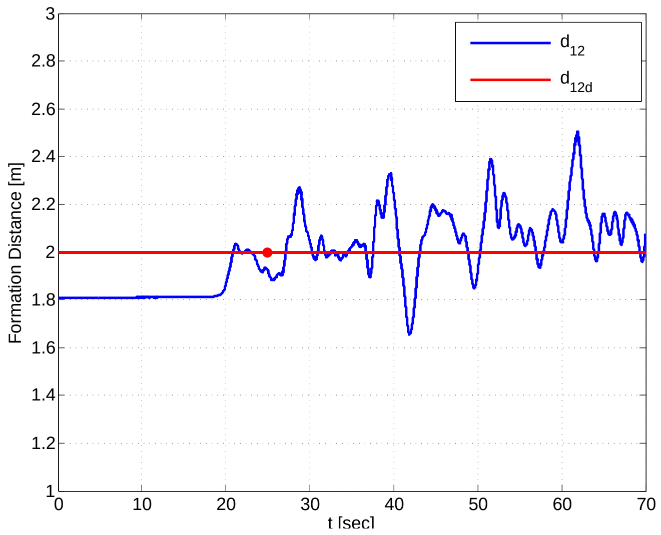
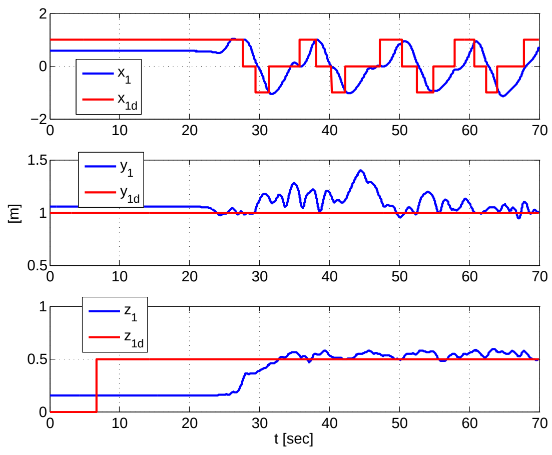
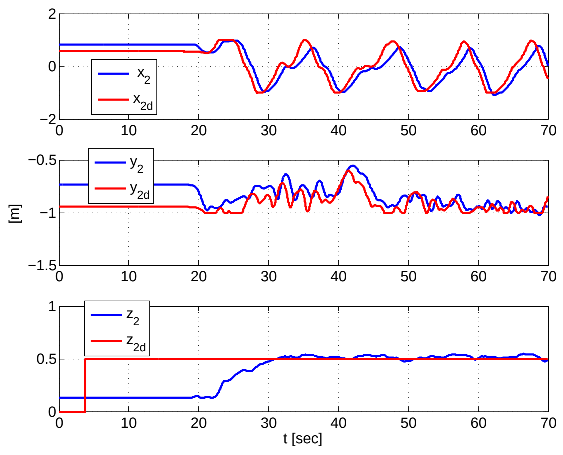


Figure 3.13: Formation distance for line path in experimental implementation

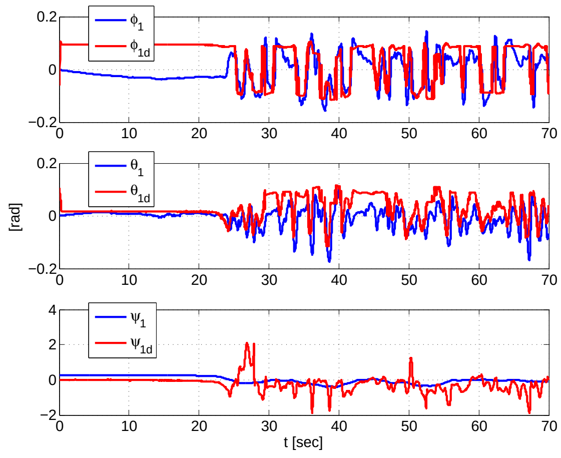


(a) Leader reference tracking



(b) Follower reference tracking

Figure 3.14: Tracking performances of agents for line path in experimental implementation


(a) Leader attitude angles

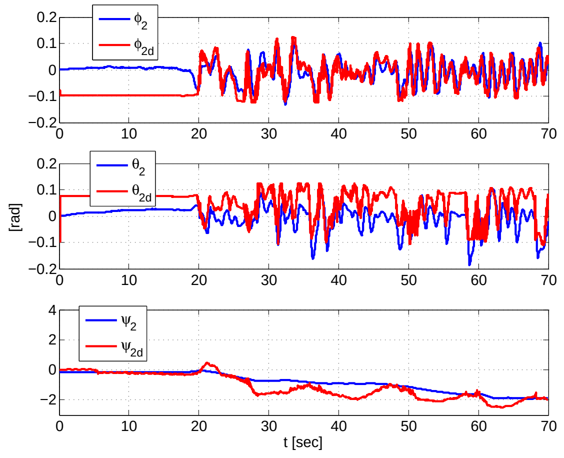
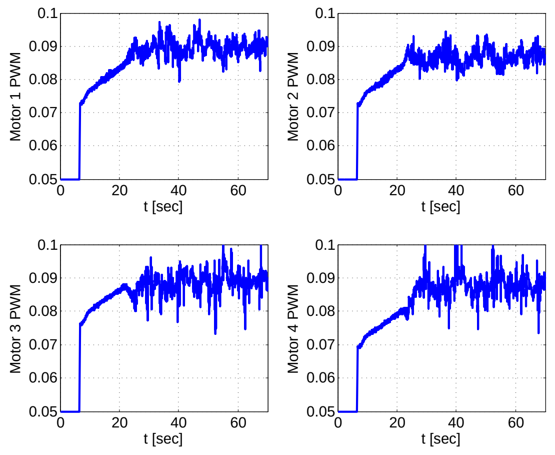

(b) Follower attitude angles

Figure 3.15: Attitude performances of agents for line path in experimental implementation


(a) Leader PWM inputs

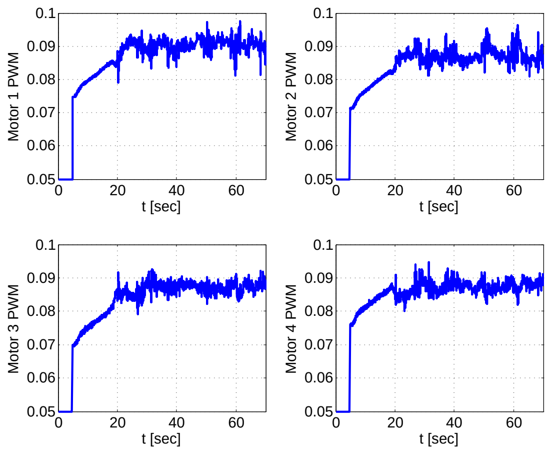

(b) Follower PWM inputs

Figure 3.16: Control inputs of agents for line path in experimental implementation

Chapter 4

On-line Identification and Adaptive Formation Control

This section focuses on on-line parameter identification (PI) of the multi-quadrotor swarm system during the formation motion. In Chapter 4, we control the multi-quadrotor system by using specified control parameters in a linear quadratic controller (LQR). On the other hand, note that in the experiment, the quadrotor system has uncertainties on some of control parameters because of their dynamic or various neglected dynamic effects on the quadrotor. Therefore, we consider that inertia has uncertainty in the simplified linear model (2.5.17) of the roll and pitch model during the motion. Furthermore, we focus on on-line identification of inertia in roll and pitch model, and then we develop a decentralized adaptive linear quadratic control (ALQR) for performing the formation task.

4.1 Motivation and Problem Definition

On-line parameter identification (PI) works between current input and past output measurements of signals which are known or detected with the help of various kinds of equipment. Afterward, by using these measurements, the estimation algorithm generates values for the unknown plant parameters at each instant of time [18]. In this chapter, we consider that the inertia parameters in the simplified model are not constant values and change during the motion. Hence, we identify this value at each instant of time during the estimation, and then develop an ALQR control scheme for performing the adaptive formation task.

In order to perform the formation objective with the adaptive parameter estimation, we use the same formation design as mentioned in Chapter 3. Therefore, it is considered that formation modeling, on-line trajectory generation and reference angle generation are taken similarly as non-adaptive design to provide condition (3.2.1) from Chapter 3. We give our formation problem as follows:

Problem 4.1.1 *[Adaptive Formation Problem] Consider a swarm S with $A_i,\ i = 1, ..., 3$ as in Figure 3.2. The subblock $S_i,\ i = 1, ..., 3$ represents real dynamics and controller of each single quadrotor agent. Equation* (2.5.17) *represents the dynamic of motion in the x-y plane, and the exact value of inertia J_ζ parameter is not known during the motion. Generate control signals $u_\zeta(t)$ $\forall t$, for each agent under the consideration of assumptions (3.2.1),(3.2.2) and (3.2.3) so that $P_i(t)$ tracks $P_{id}(t),\ i = 1, ..., 3$ for maintaining distance and triangular formation, asymptotically.*

Afterwards, considering *Problem* 4.1.1 and by using a designed non-adaptive formation structure as discussed in Chapter 3, we develop an adaptive linear quadratic control

(ALQR) scheme for the autonomous formation experiment and its simulation. Therefore, the on-line identification procedure consists of three steps, namely, setting unknown parameters in form of parametric model, designing an adaptive estimation algorithm, and establishing parameter convergence and stability. These steps will be used for on-line PI algorithms in the next section.

4.2 On-line Parameter Identification

Following [18], we first a design static parametric model (SPM), and then a least squares estimation algorithm to find the unknown parameters in the attitude behavior of the quadrotor system. The roll and pitch dynamic equation (2.5.17) is written in Laplace form as

$$s^2\zeta = \frac{L}{J_\zeta}\Delta T = \frac{L}{J_\zeta}\left(2K\frac{w}{s+w}u_\zeta\right), \tag{4.2.1}$$

where ζ represents measurable outputs of system which are pitch/roll angles and u_ζ represents known inputs of pitch/roll dynamic models. ΔT_ζ represents the net changes for generated thrust forces as pitch/roll inputs.

Note that ζ is measurable by using inertial measurement unit (IMU) on the quadrotor, and u_ζ represents inputs calculated by adaptive controller. Also, L, K and w are specified constant parameters for pitch and roll dynamics, respectively. J_ζ is taken as a constant parameter in chapter 3, however, we consider that this parameter has uncertain behavior in attitude dynamics during the motion because its inertial resistance is supposed to change at different speeds. In implementation, we do not deal with speed control, and the quadrotor system works within proper speed range during the task; thus, it is considered that J_ζ is an uncertain parameter in dynamics to identify on-line as informed follows.

4.2.1 Parametric Model

We can express (4.2.1) in form of a static parametric model (SPM). Thus, we filter each side by the stable filter $\frac{1}{(s+\lambda)^2}$, $\lambda > 0$, as follows. Then, we obtain

$$\frac{s^2}{(s+\lambda)^2}\zeta = \frac{1}{J_\zeta}\frac{2KL}{(s+\lambda)^2}\frac{w}{(s+w)}[u_\zeta], \tag{4.2.2}$$

where ζ and u_ζ represent output and input of pitch and roll model for each agent, respectively. Then, we write a parametric model for (4.2.2) as

$$z = \theta^*\phi, \tag{4.2.3}$$

$$z = \frac{s^2}{(s+\lambda)^2}\zeta, \quad \theta^* = [\frac{1}{J_\zeta}], \quad \phi = [\frac{2KL}{(s+\lambda)^2}\frac{w}{(s+w)}(u_\zeta)].$$

Note here that θ^* represents the uncertain system parameter.

4.2.2 Parameter Estimation using Least Squares

We apply recursive the least squares algorithm [18] to the parametric model (4.2.3) as

$$\begin{aligned}\dot{\hat{\theta}}(t) &= P(t)\epsilon(t)\phi(t), \quad \hat{\theta}(0) = \hat{\theta}_0, \\ \dot{P}(t) &= \beta P(t) - \frac{\phi^2(t)}{m^2(t)}P^2(t), \quad P(0) = P_0, \\ \epsilon(t) &= \frac{z(t) - \hat{z}(t)}{m^2(t)}, \quad m^2(t) = 1 + \phi^2(t),\end{aligned} \tag{4.2.4}$$

where $P \in \mathbb{R}$ is the covariance and $P > 0$, m is the normalizing signal, $\phi \in \mathbb{R}$ is the regressor signal and ϵ is the estimation error. The estimation model is given by

$$\hat{z}(t) = \hat{\theta}(t)\phi(t), \tag{4.2.5}$$

where $\hat{\theta}(t)$ is the estimated parameter at time instant of time t. Stability features of this estimation algorithm are given in the following proposition.

4.2.3 Stability and Convergence

Considering parameter identification of the uncertain parameter, according to Theorem 3.7.1 of [18], the estimation algorithm (4.2.4)-(4.2.5) is applied to the pitch and roll dynamics (4.2.1). If $\frac{\phi}{m}$ is PE then the recursive LS algorithm with forgetting factor (4.2.4) guarantee that $P, P^{-1} \in \mathcal{L}_\infty$ and that $\theta(t) \to \theta^*$ as $t \to \infty$. The convergence of $\theta(t) \to \theta^*$ is exponential when forgetting factor $\beta > 0$.

We should emphasize that for the least square algorithm signal $\phi(t)$, which is the control input u_ζ itself in this specific case, to be sufficiently rich. However, this case may not always be guaranteed to converge the parameter values to its true value in the least squares algorithm. Here, the control law has to achieve formation objectives. Therefore, a different approach is used to guarantee the control objective by using an on-line estimated θ. It is considered that if the value of θ exceeds its limitation which is specified in adaptive controller design during the estimation, θ is assumed in its ideal value specified by Quanser inc. as seen in Figure 3.1. Otherwise, the estimated θ value is used for the adaptive control scheme at each instant time.

4.3 Adaptive Linear Quadratic Control (ALQR) Design

In this section, we design an adaptive linear quadratic controller scheme for each quadrotor's pitch and roll models as seen in the single-quadrotor control scheme diagram in Figure 4.1. After the on-line parameter estimation of inertia in pitch and roll dynamics, we design an adaptive control law to control quadrotor for x-y motion. The overall control structure is considered in two parts, as seen in Figure 4.1, since we consider the pitch and roll dynamics separately for x-y motion. Furthermore, although we design an adaptive control scheme for the pitch and roll model, we use same control structures for the altitude and yaw model as stated in Chapter 3.

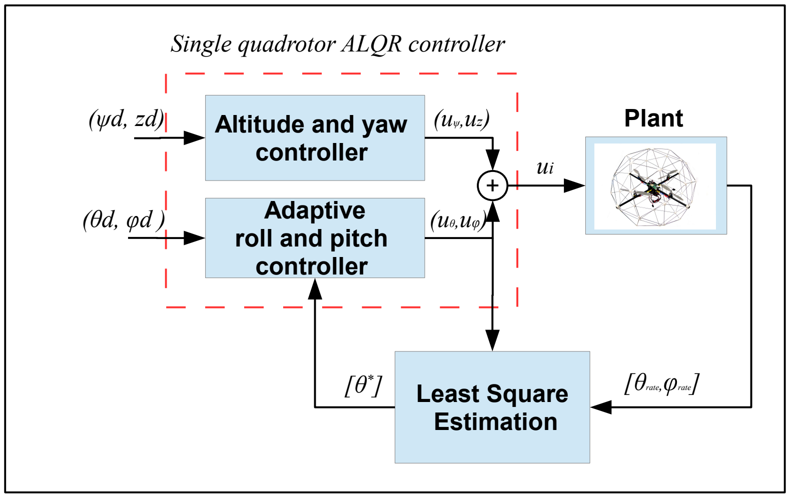

Single quadrotor ALQR controller
(ψd, zd)
Altitude and yaw controller
(uψ,uz)
Plant
ui
(θd, φd)
Adaptive roll and pitch controller
(uθ,uφ)
[θ*]
Least Square Estimation
[θrate,φrate]

Figure 4.1: Adaptive linear quadratic control (ALQR) scheme for an individual quadrotor agent

Thus, considering the linearized dynamic equation (2.5.17), we present an adaptive linear quadratic controller (ALQR) design to generate effective control inputs of pitch and roll model which are u_θ and u_ϕ. In order to design an adaptive linear quadratic control scheme, generic state-space form of linearized dynamic representation are obtained with respect to on-line parameter estimation from (2.5.21) and (4.2.5) as

$$\dot{x}_\zeta = \hat{A}_\zeta x_\zeta + B_\zeta u_\zeta. \tag{4.3.1}$$

where $\hat{A}_\zeta$ is a 4x4 on-line estimated matrix and B_ζ is a 4x1 matrix, and for minimizing performance measurement of the system, we can use the cost function as

$$J = \int (x^T Q x + \lambda u^2) dt, \tag{4.3.2}$$

where Q is a 4x4 positive semidefinite matrix and λ is a positive scalar.

Then, in order to find stable adaptive feedback gain, a state-feedback control law is written as

$$u = -\hat{K}_c x. \tag{4.3.3}$$

$\hat{K}_c$ is an 1x4 on-line calculated gain matrix obtained as

$$\hat{K}_c = \lambda^{-1} B_\zeta^T \hat{P}, \tag{4.3.4}$$

where $\hat{P}$ is a 4x4 auxiliary matrix calculated at each instant of time by solving the Riccati equation as

$$\hat{A}_\zeta^T \hat{P} + \hat{P}\hat{A}_\zeta - \hat{P}B_\zeta \lambda^{-1} B_\zeta^T \hat{P} + I = 0, \quad \hat{P} = \hat{P}^T > 0. \tag{4.3.5}$$

4.4 Simulation and Experimental Studies

This section present the results of real-time adaptive formation implementation and simulation for different references. The difference is that for the pitch and roll model, we apply on-line parametric estimation for inertial change as stated in Section 4.2, and then depending on this estimation, we use a pre-prepared look-up gain table which is calculated off-line by the LQR control scheme for each estimated value of inertia. Therefore, during the motion, to calculate ALQR control gain $\hat{K}_c$, this table is used based on estimation value of $\theta(t)$ at any time.

At first, for two reference cases we employ an adaptive formation simulation with a specified noise effect by MATLAB *Simulink*® for the quadrotor system as designed in Chapter 3. Furthermore, all parameters are taken from table 3.1 except for inertia value of pitch and roll dynamics. Then, the experiment is applied to the adaptive formation control design depending on earlier formation design.

In the implementation, in order to design the adaptive linear quadratic controller (ALQR), we use input and output data from (4.2.3). Although they are filtered for the estimation model by a low-pass filter, the noise effect is not canceled over the data. For this reason, we use a Butterworth low-pass filter to cancel the noise effect on the data of estimation model. In order to find the ideal filter, we use the low-pass Butterworth filter gain equation

$$|H(jw)| = \frac{1}{\sqrt{1 + (\frac{w}{w_c})^{2n}}}, \tag{4.4.1}$$

where w_c is cutoff frequency and n is filter order.

Therefore, from (4.4.1) and ideal cutoff gain, for randomly chosen filter order, we can obtain cutoff frequency w_c=15 as seen in Figure 4.2. Afterwards, we use this filter before the estimation algorithm for each quadrotor agent in the pitch and roll models as seen in example Figure 4.3a and 4.3b.

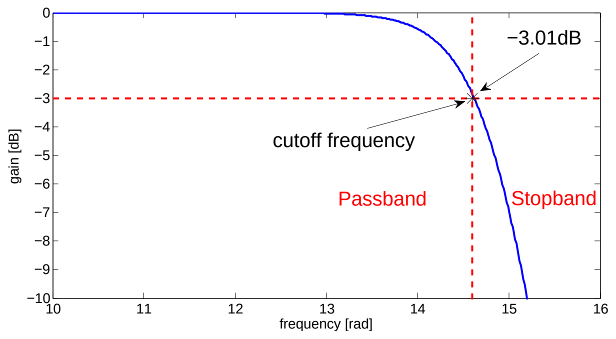
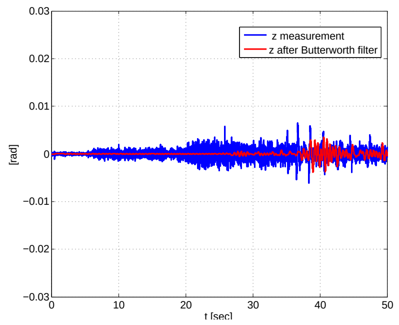
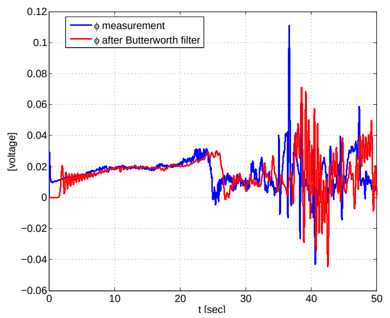


Figure 4.2: Butterworth filter gain

In the implementations, as the recursive least square algorithm parameters, we particularly use forgetting factor $\beta = 0.1$, covariance $P_0 = 10^5$, and initial theta value $\theta_0 = 10$, and



(a) z filtering



(b) ϕ filtering

Figure 4.3: Example filtering for z and ϕ estimation dynamics in roll using Butterworth filter

in experimental test, adaptive controller works within estimation parameters' limitation between $\theta = \frac{1}{J_\zeta} = 10 - 50$ using a look-up gain table which is prepared before the test for each θ values of the limitation. Otherwise, if θ is exceeded its limitation, it switches the ideal gain value for the specified theta value $\theta = 33$.

From line path results of next Section 4.4.1 and 4.4.2, by using mean square error calculation (3.6.3) and the formation distance errors of experiment and simulation, we can obtain formation performance of multi-quadrotor swarm system. Thus, for simulation and experiment, formation mean square error are demonstrated in table (4.1).

4.4.1 Simulation Results

This subsection presents simulation results of adaptive formation control as shown in Figures 4.4-4.13. It is clearly seen that after the settling of parameter estimation adaptive controller works smoother than non-adaptive case. However, as seen in Figure 4.5 and 4.10c, adaptive controller has a bigger formation distance errors and oscillations before

Table 4.1: Formation mean squared error performances by ALQR control scheme

Mean squared error	Simulation			Experimental	
	Leader	Follower 1	Follower 2	Leader	Follower 1
Overall term	2.005	0.0112	0.0289	1.65	0.021
After the settling	0.471	0.0069	0.0062	1.55	0.020

settling steady-state formation error because parameter estimation convergence takes time for spiral path at around 40 [sec] and line path at 5 [sec] as seen in Figure 4.6 and 4.11.

When sensor noises are applied in simulation, parameter estimation is affected to converge to ideal value, especially, in line path since its inputs have sufficiently rich problem. However, as seen in Figure 4.4, 4.5 and 4.10, our algorithm works satisfactorily and then all agents track their references. Also each follower maintain its desired distance and the desired formation shape with small error as seen in Figure 4.5 and 4.10c. Note that all control input are suit able for real-time implementations.

4.4.2 Experimental Results

Adaptive Formation control experimental results are given in Figure 4.14 - 4.18 for line path. As they are shown, the two-level distributed control structure is able to tackle the formation problem described in Section 4.1. For testing our algorithm, as seen in Figure 4.15, our algorithm works satisfactorily for maintaining the desired rigid distance in real-time environment. It is clearly seen that adaptive controller works smoother than non-adaptive case after the quadrotors hovering. As seen in Figure 4.16, it expected that adaptive controller should have a bigger formation distance errors and oscillations before settling steady-state formation error, however, because of using lookup table, these errors are compensated.

Both of the agents track their references as seen in Figure 4.18 after the system ready for hovering at 25 [sec]. Follower maintains its desired distance with small error as seen in 4.16. Note that the orientation of formation does not change during the motion, and agents track attitude angles as seen in Figures 4.19.

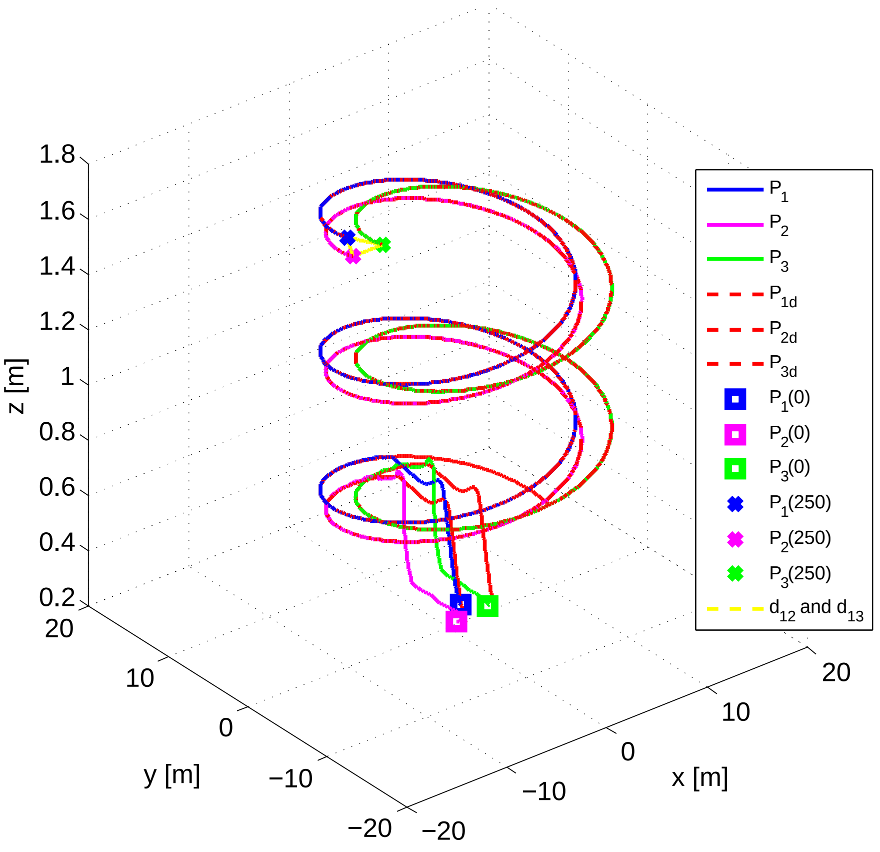


Figure 4.4: Agents' motions for spiral path with noise effect

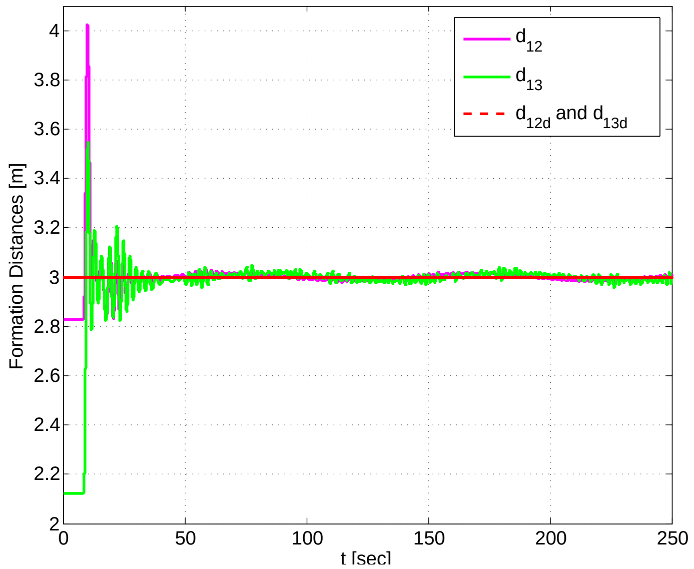


Figure 4.5: Formation distances among leader and followers for spiral path with noise effect

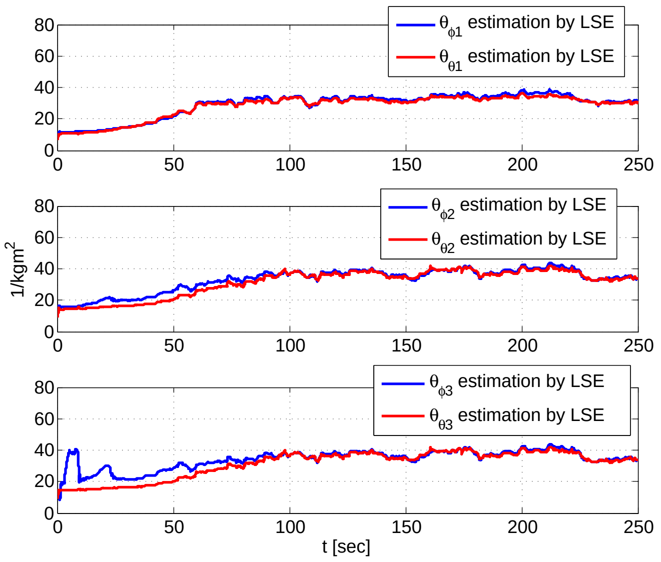
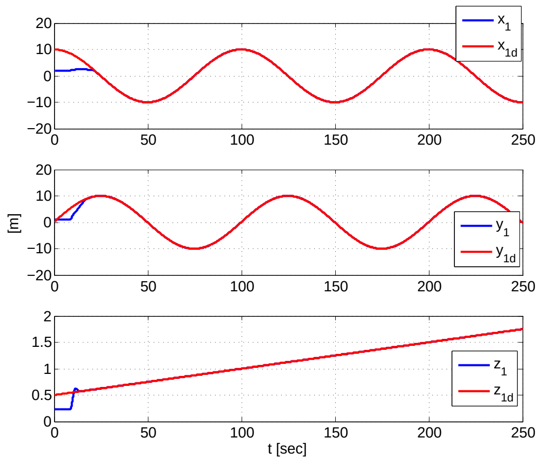
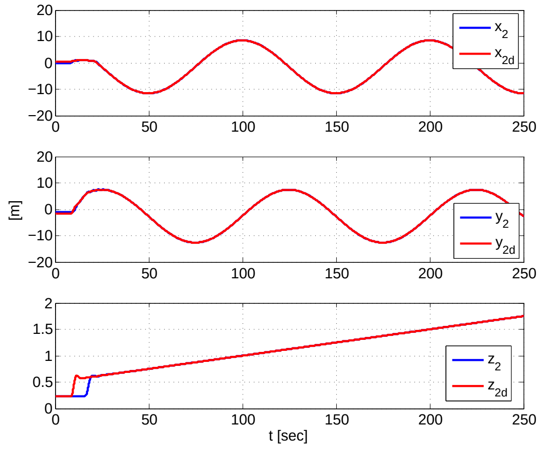


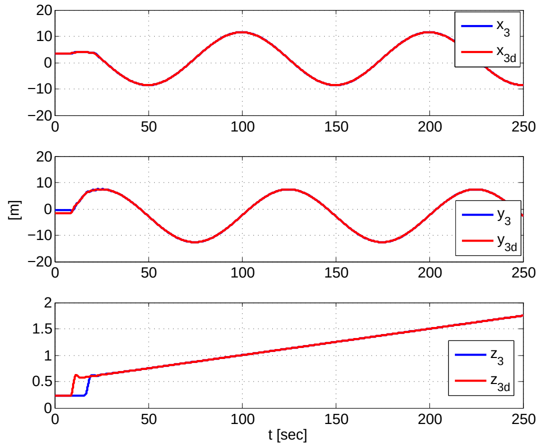
Figure 4.6: Parameter estimation for pitch and roll motions of leader and followers



(a) Leader reference tracking

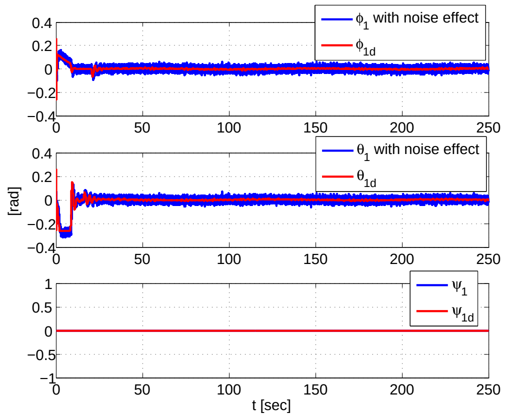


(b) Follower 1 reference tracking

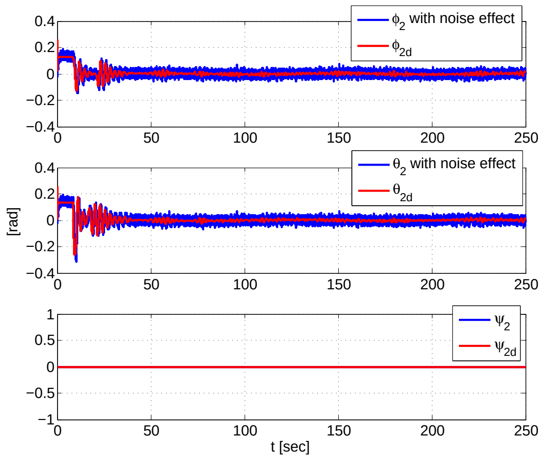


(c) Follower 2 reference tracking

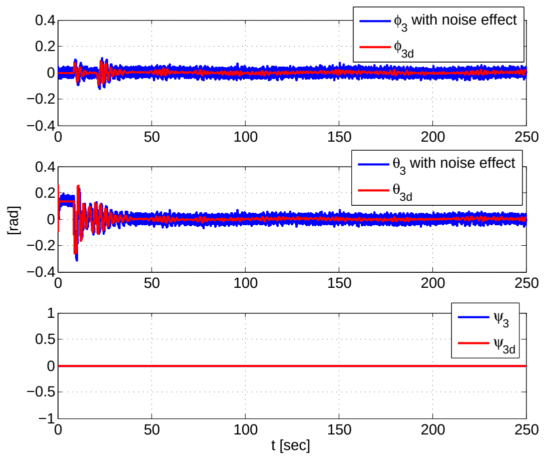
Figure 4.7: Tracking performances of agents for spiral path with noise effect



(a) Leader attitude angles

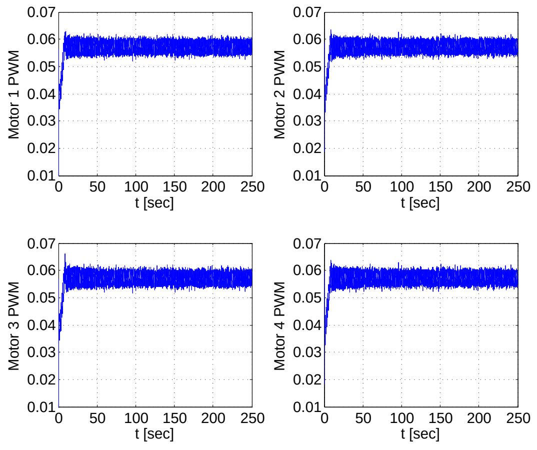


(b) Follower 1 attitude angles

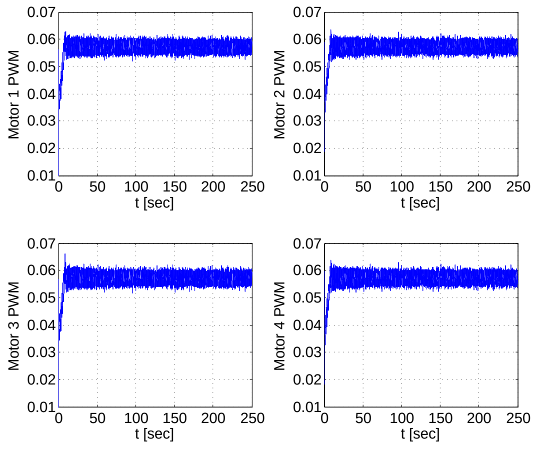


(c) Follower 2 atitude angles

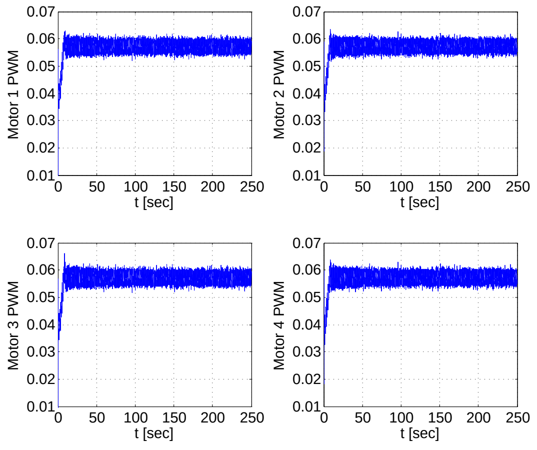
Figure 4.8: Attitude performances of agents for spiral path with noise effect.



(a) Leader PWM inputs

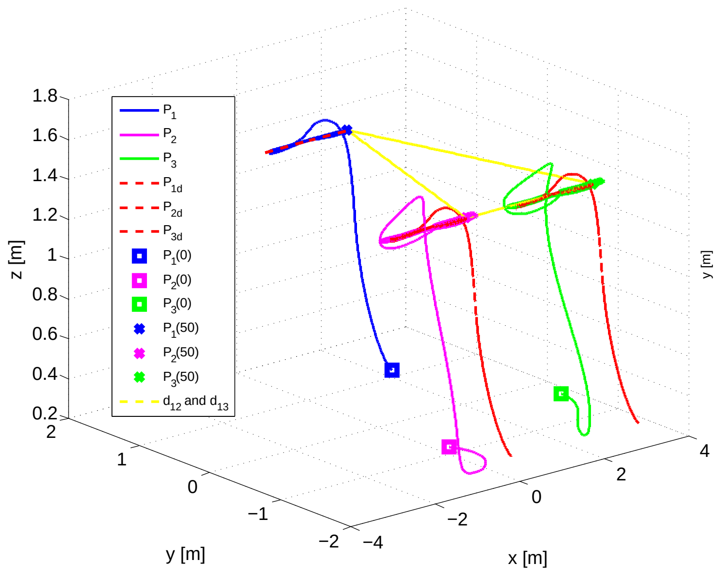


(b) Follower 1 PWM inputs

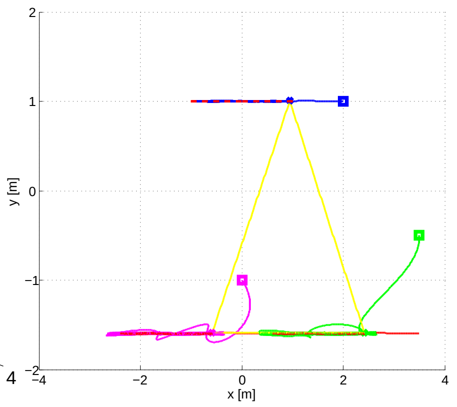


(c) Follower 2 PWM inputs

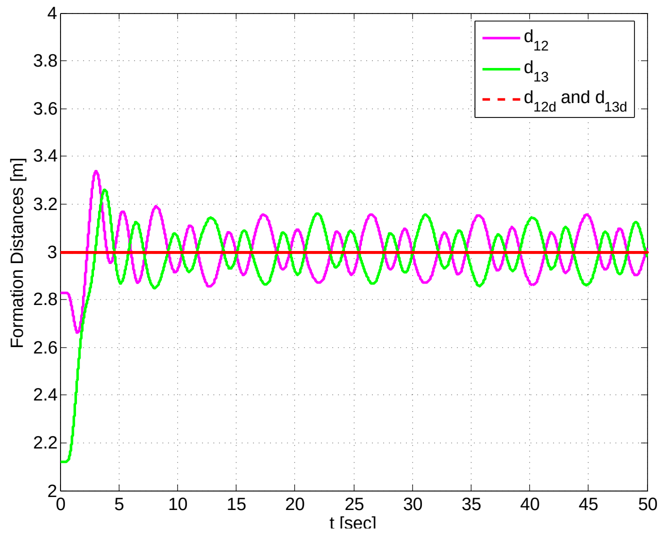
Figure 4.9: Control inputs of agents for spiral path with noise effect.



(a) Pathways of agents



(b) Pathways in x-y motion



(c) Formation distances among leader and followers

Figure 4.10: Agents' motions and formation distances for line path with noise effect.

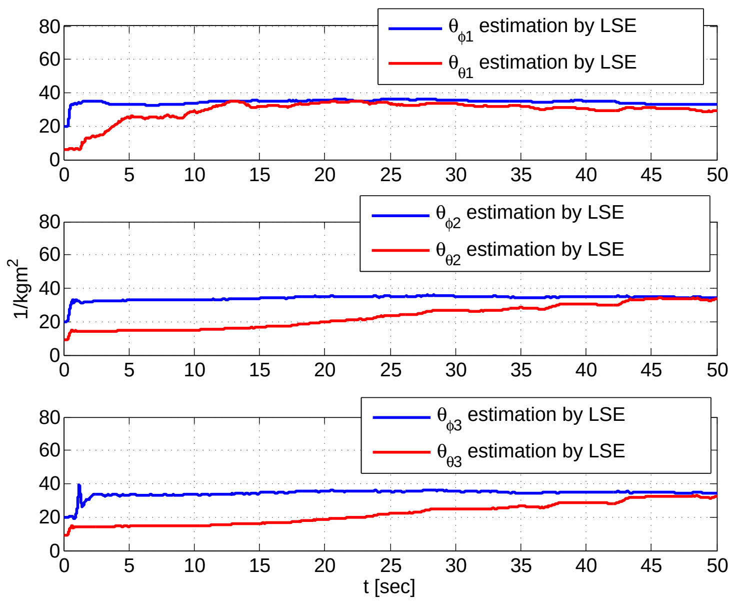
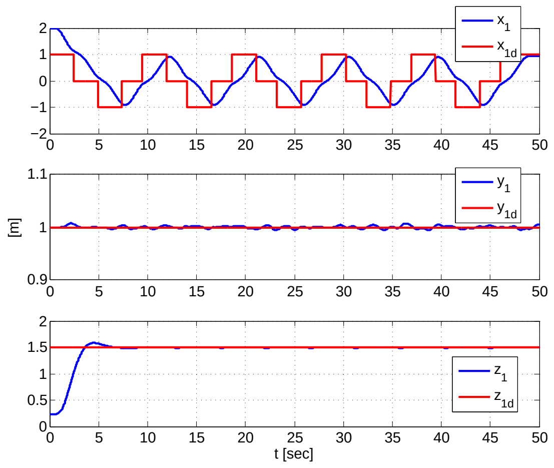
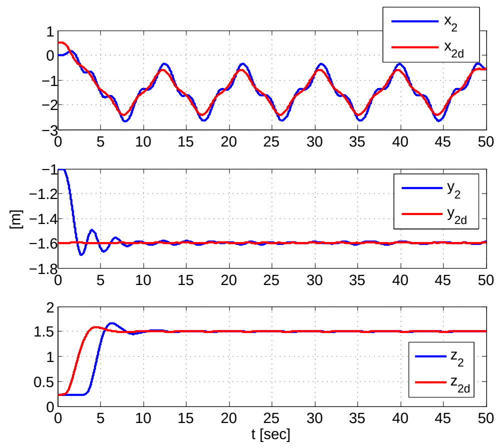


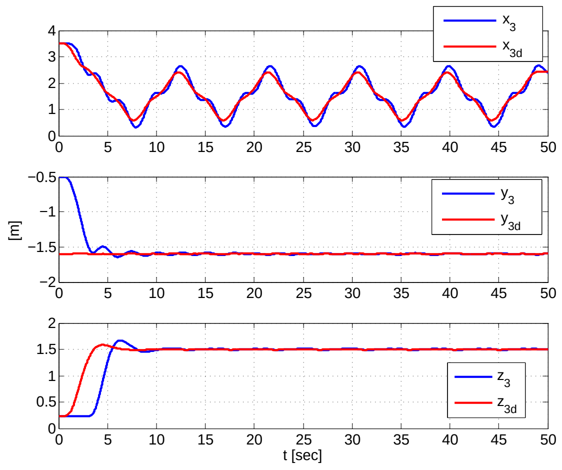
Figure 4.11: Parameter estimation for pitch and roll motions of leader and followers



(a) Leader reference tracking

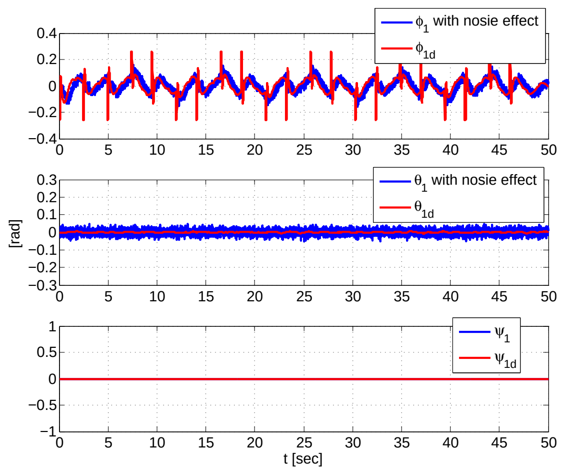


(b) Follower 1 reference tracking

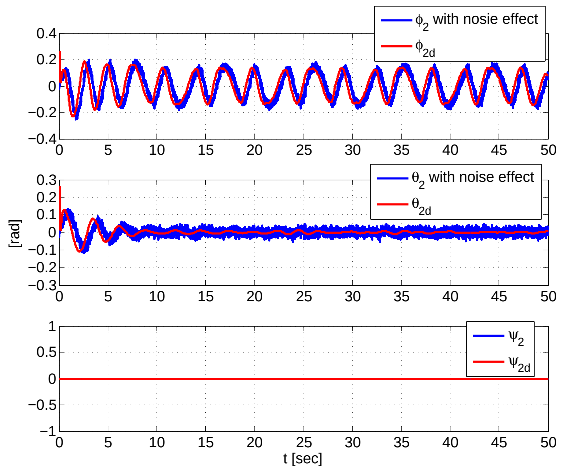


(c) Follower 2 reference tracking

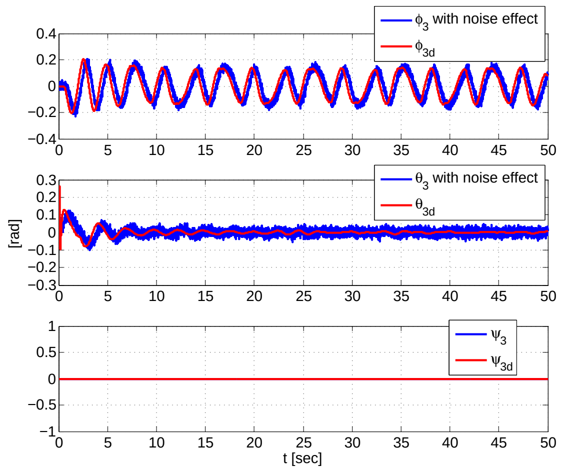
Figure 4.12: Tracking performances of agents for line path with noise effect



(a) Leader attitude angles

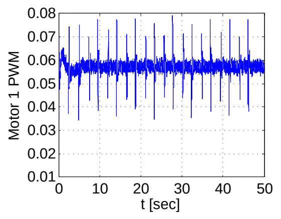


(b) Follower 1 attitude angles

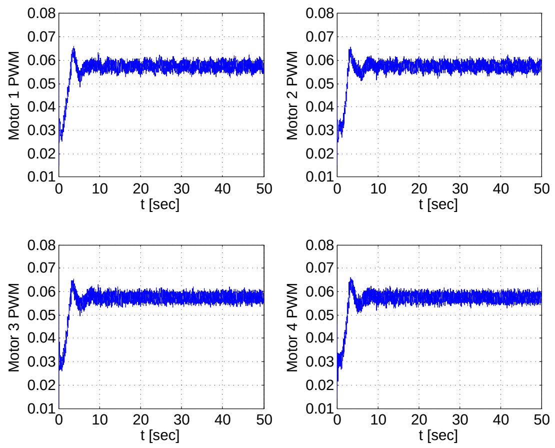


(c) Follower 2 attitude angles

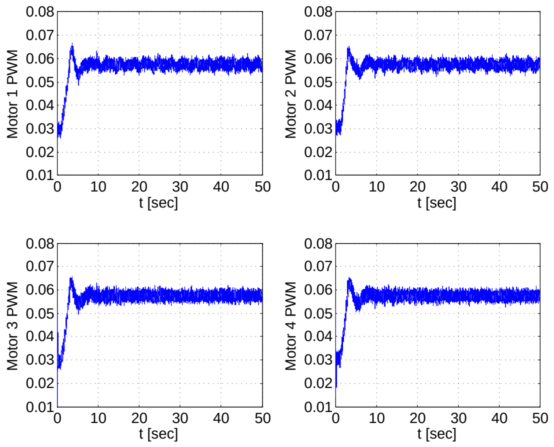
Figure 4.13: Attitude performances of agents for line path with noise effect



(a) Leader PWM inputs

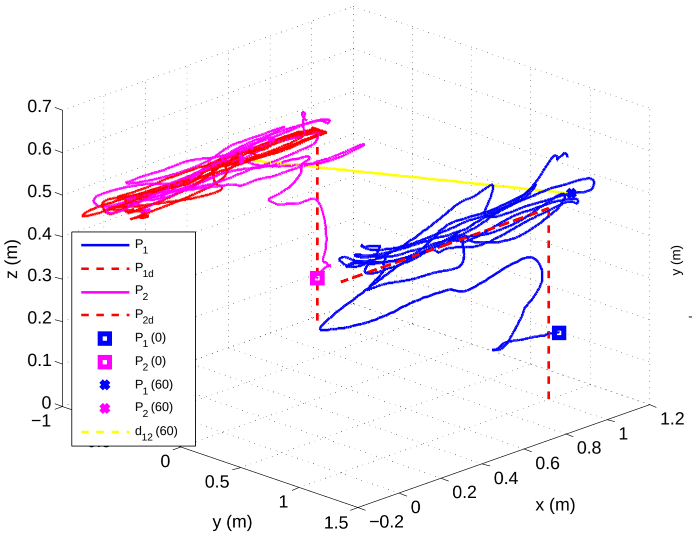


(b) Follower 1 PWM inputs

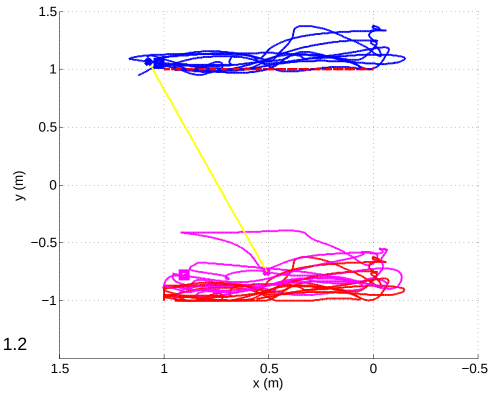


(c) Follower 2 PWM inputs

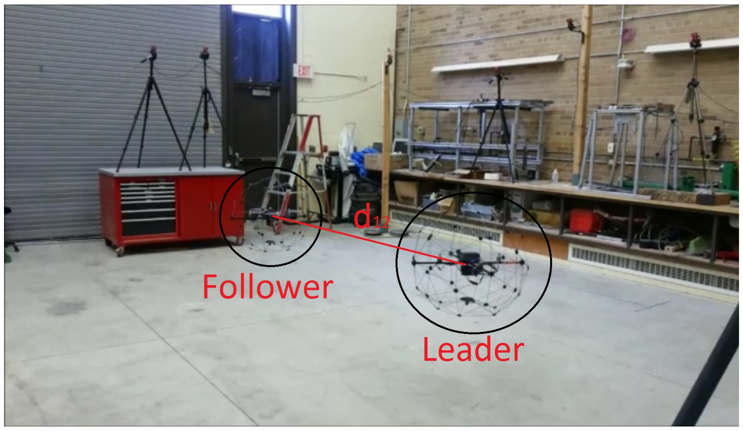
Figure 4.14: Control inputs of agents for line path



(a) Pathways of agents



(b) Pathways in x-y motion



(c) Experimental view of leader and follower during the task

Figure 4.15: Agents' motions for line path in experimental implementation

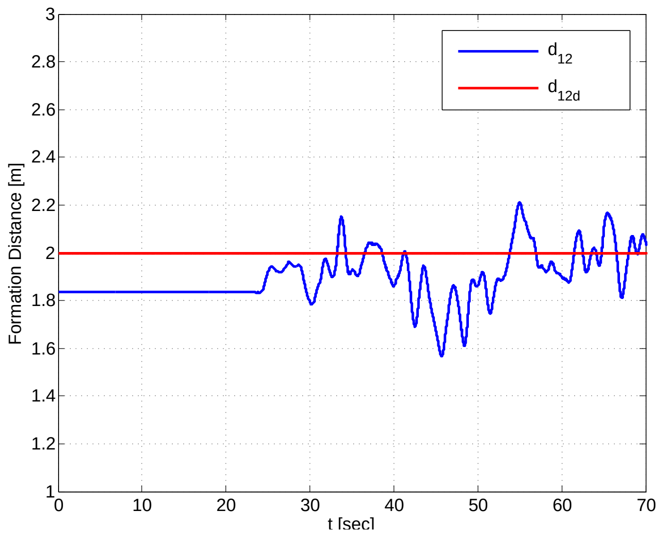


Figure 4.16: Formation distance for line path in experimental implementation

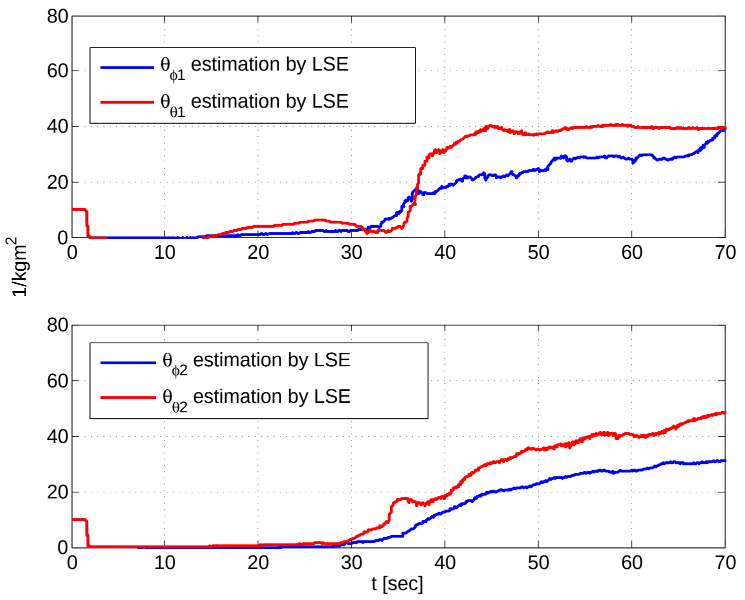
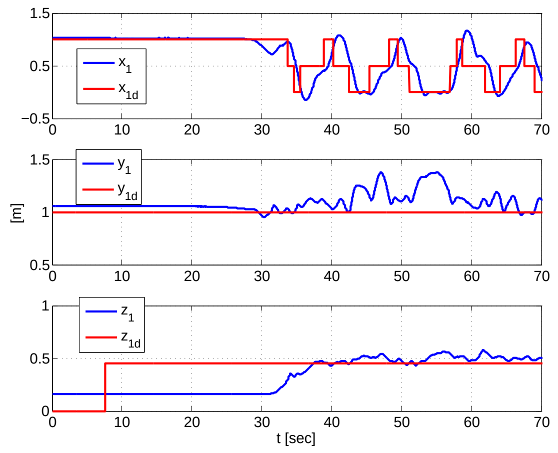
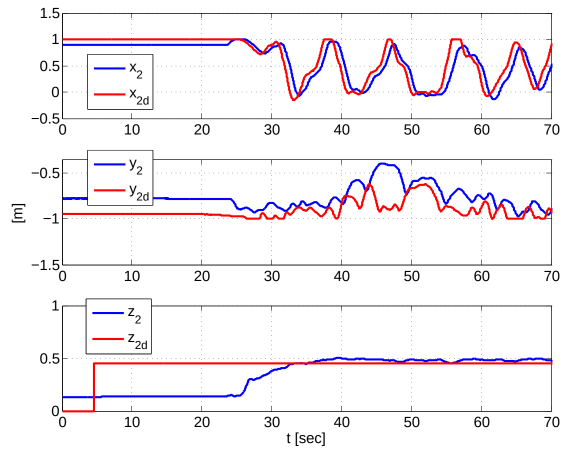


Figure 4.17: Parameter estimation for pitch and roll motions of leader and follower.

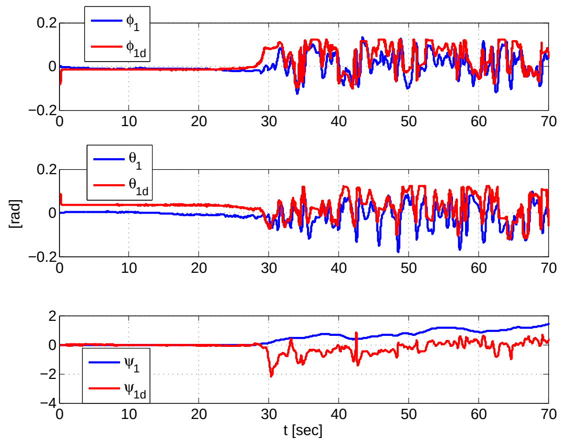


(a) Leader reference tracking

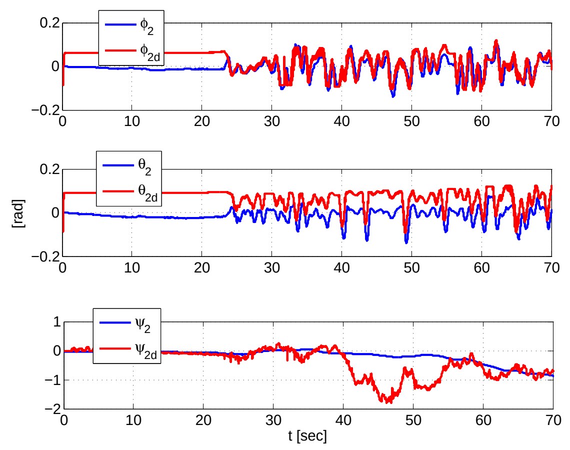


(b) Follower referene tracking

Figure 4.18: Tracking performance of agents for line path in experimental implementation

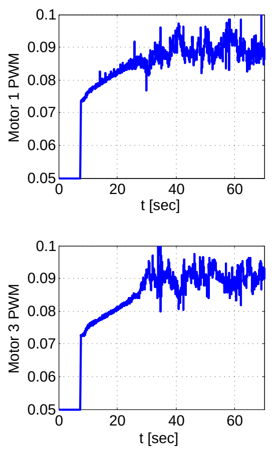


(a) Leader attitude angles

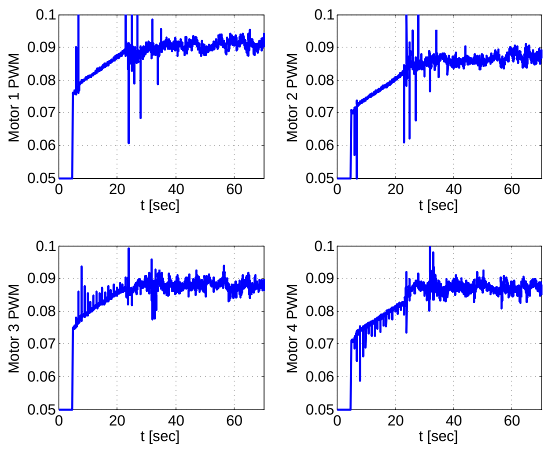


(b) Follower attitude angles

Figure 4.19: Attitude performances of agents for line path in experimental implementation



(a) Leader PWM inputs



(b) Follower PWM inputs

Figure 4.20: Control inputs of agents for line path in experimental implementation

Chapter 5

Localization Enhancement Using Single-View Distance Estimation

In an attempt to acquire more advantages in localization of the multi-quadrotor swarm, different localization methods can be used by the leader agent to estimate its absolute (or global) position or by the follower agents to estimate relative positions of the agents they follow. Aiming and enhancing localization accuracy in an easily implementable based on single-view distance estimation [4]. Such a methodology will be useful for the multi-quadrotor system indoor localization problems such as restricted workspace because the main indoor localization systems such as Optitrack have limited work space area. In [4], single-view (camera) based distance estimation method is used for inter-agent distance estimation and formation control of a ground robotic swarm. Here, we develop a similar methodology for indoor aerial vehicle networks, particularly our multi-quadrotor system.

5.1 Camera Modeling and Quantization

In this section, we assume a conceptual setting where each agent is equipped with a single camera [36] at a suitable place in the lower part of the quadrotor as illustrated in Figure 5.1, and there are two specified landmarks on the ground. It is assumed that the landmarks are in the field of view during the task. Here, by geometry of a single perspective camera, we design the localization system to estimate distance and direction of the landmark based on the studies in [4, 37].

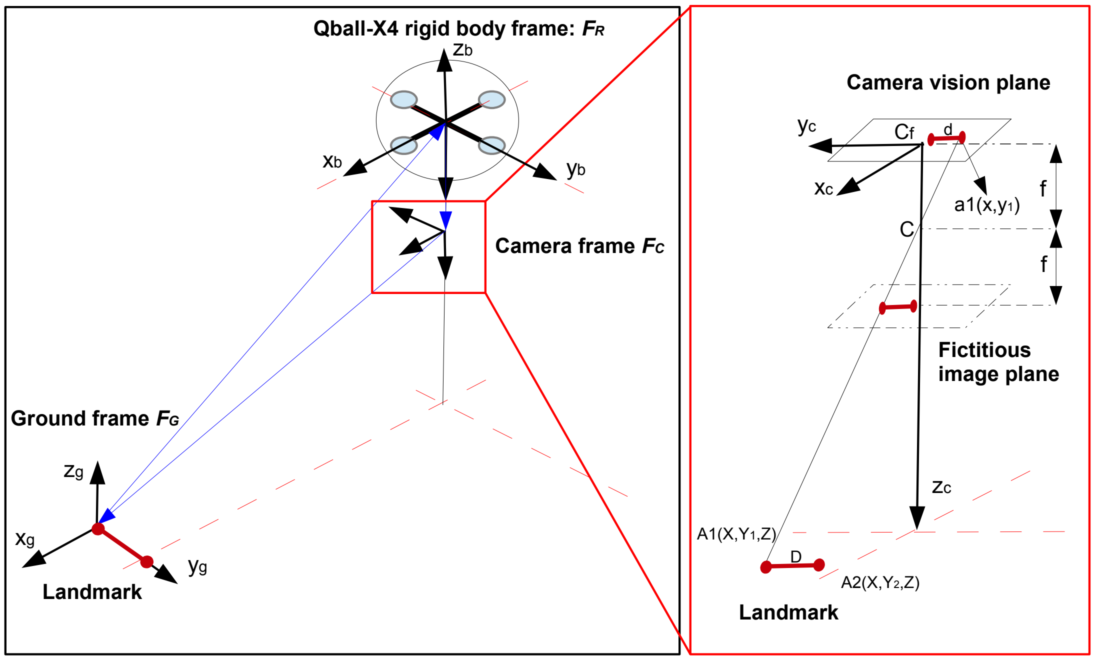


Figure 5.1: Modeling of a single-view camera vision and frames

As seen in Figure 5.1, vision geometry of camera has optical center C as a camera frame center, C_f focus point, the focal length f and and the principal axes parallel to the z-axis [4, 37]. Here, camera vision plane is parallel to fictitious image plane, and optical center is in the center of this parallel planes. Therefore, we obtain the position relation between particular position of landmark A_1 and camera vision position a_1, and the camera calibration matrix using these components and their geometry as

$$\frac{f}{Z} = \frac{x}{X} = \frac{y_1}{Y_1}. \tag{5.1.1}$$

Then the central projection is written as

$$Z \begin{bmatrix} x \\ y_1 \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y_1 \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y_1 \\ Z \\ 1 \end{bmatrix}, \tag{5.1.2}$$

where P is the camera projection matrix [4, 37].

In order to calculate small difference (p_x, p_y) in the match of the frames between real camera frame and assumed camera frame in Z-axis, we use

$$x = \frac{fX}{Z} + p_x, \quad y_1 = \frac{fY_1}{Z} + p_y, \tag{5.1.3}$$

then we write the central projection matrix as

$$Z \begin{bmatrix} x \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} K' & | & 0 \end{bmatrix} \begin{bmatrix} X \\ Y_1 \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y_1 \\ Z \\ 1 \end{bmatrix}, \tag{5.1.4}$$

where K' is the camera calibration matrix 3x3 [4, 37].

Moreover, adding the number of pixels per unit distance (m_x, m_y), in images coordinate, pixel values (x, y_1) are converted to $(\bar{x}, \bar{y}_1)$, and in order to quantize the pixels of the camera [4], we obtain

$$\begin{bmatrix} \bar{x} \\ \bar{y}_1 \\ 1 \end{bmatrix} = q\left(Z^{-1}\begin{bmatrix} K & \mid & 0 \end{bmatrix} \begin{bmatrix} X \\ Y_1 \\ Z \\ 1 \end{bmatrix} \right) = q\left(Z^{-1}\begin{bmatrix} fm_x & 0 & p_x m_x & 0 \\ 0 & fm_y & p_y m_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y_1 \\ Z \\ 1 \end{bmatrix} \right), \quad (5.1.5)$$

where q is quantization operator, and

$$K = \begin{bmatrix} fm_x & o & p_x m_x \\ 0 & fm_y & p_y m_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_x & o & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (5.1.6)$$

By considering non-rectangle pixels, skew parameter is added in K as

$$K = \begin{bmatrix} a_x & o & x_0 \\ s & a_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.1.7)$$

Finally, we obtain

$$\begin{bmatrix} \bar{x} \\ \bar{y}_1 \\ 1 \end{bmatrix} = q\left(Z^{-1}\begin{bmatrix} K & \mid & 0 \end{bmatrix} \begin{bmatrix} X \\ Y_1 \\ Z \\ 1 \end{bmatrix} \right) = q\left(Z^{-1}\begin{bmatrix} a_x & 0 & x_0 & 0 \\ s & a_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y_1 \\ Z \\ 1 \end{bmatrix} \right), \quad (5.1.8)$$

5.2 Single-View Distance Based Position Estimation

Using equation (5.1.4), a single camera and a specified landmark, we can calculate x-y direction of quadrotor assuming the z distance is known; however, we can estimate z-distance if we have two specified landmarks and we know the distance between them, using triangular similarities. Here, it is considered that we have a camera set as shown in Figure 5.2 and two detectable objects placed at $A_1(X, Y_1, Z)$ and $A_2(X, Y_2, Z)$. The goal is self-localization of the quadrotor agent of interest using the single camera mounted on the lower part of it as depicted in Figure 5.1. From equation (5.1.8), we obtain between pixel coordinate to global coordinate as

$$\begin{aligned} \bar{y}_i &= q(Z^{-1}(a_y Y_i + y_0 Z + sX)), \qquad i = 1, 2. \\ \bar{x} &= q(Z^{-1}(a_x X + x_0 Z)), \end{aligned} \tag{5.2.1}$$

where $(\bar{x}, \bar{y}_1)$ and $(\bar{x}, \bar{y}_2)$ pixel coordinates of the images of A_1 and A_2. If we consider the quantization remainder on the estimation, let $\tilde{q} : \mathbb{R} \longrightarrow [0, 1)$ be the quantization remainder operator which is described as $\tilde{\xi} = \xi - q(\xi)$ for any $\xi \in \mathbb{R}$ [4], we have as

$$\begin{aligned} \tilde{y}_i &= \tilde{q}(Z^{-1}(a_y Y_i + y_0 Z + sX)), \qquad i = 1, 2. \\ \tilde{x} &= \tilde{q}(Z^{-1}(a_x X + x_0 Z)). \end{aligned} \tag{5.2.2}$$

Then, we obtain

$$\begin{aligned} \bar{y}_1 - \bar{y}_2 &= Z^{-1} a_y (Y_1 - Y_2) - (\tilde{y}_1 + \tilde{y}_2), \qquad i = 1, 2. \\ \bar{x} &= Z^{-1}(a_x X + x_0 Z) - (\tilde{x}), \end{aligned} \tag{5.2.3}$$

where $\bar{d} = \bar{y}_1 - \bar{y}_2$, $\tilde{d} = \tilde{y}_1 - \tilde{y}_2$ and $D = Y_1 - Y_2$.

Using the prior information of distance D between landmarks and calculation of pixel coordinate distance $\bar{d}$ on vision plane as seen in Figure 5.1, we obtain

$$Z = \frac{a_y D}{\bar{d} + \tilde{d}}. \tag{5.2.4}$$

Afterward, from (5.2.4) and (5.2.3), we calculate

$$X = \frac{a_y D(\bar{x} - x_0 + \tilde{x})}{a_x(\bar{d} + \tilde{d})}. \tag{5.2.5}$$

After calculation of Z and X, form (5.2.1) and (5.2.2), we obtain the Y_1 as

$$Y_1 = \frac{Z(\bar{y}_1 + \tilde{y}_1 - x_0) - sX}{a_y}. \tag{5.2.6}$$

Thus, we obtain position information of chosen landmark $A_1 = (X, Y_1, Z)$ for camera frame. Here, we assume that A_1 is the global reference for the design. Therefore, using generic Euler transformation between frames, we can obtain global position of the particular quadrotor agent's fixed-body according to the global position reference landmark A_1 as illustrated in Figure 5.1.

(a) Firefly camera

(b) Fujifilm lens

Figure 5.2: Camera and lens set for the design [36, 38]

In our design, we have a Pointgrey firefly camera (Model:FMVU-13S2C-CS) and Fujifilm lens (Model:YV3.3x15SA-2) with adjustable focal length $f = 15 - 50mm$, 1328x1048 pixels and pixel density $m_x = 1/(3.614\mu m)$, $m_x = 1/(3.435\mu m)$. In order to designate the limitation of vision with respect to these parameters, for focal length $f = 15mm$, note that

$$a_x = fm_x = 4150, \quad a_y = fm_y = 4366 \tag{5.2.7}$$

$$0 < \bar{x} \leqslant 1328, \quad 0 < \bar{d} \leqslant 1048 \tag{5.2.8}$$

The limitation on the operation region of the quadrotor to be able to detect the landmarks is related to field of view (FOV), and for our camera and lens set, FOV is given as $18°08'$x$13°34'$.

During the model design, we assume that heading directions of sensing camera is toward the sign in the direction of x_c to calculate more accurate position information. In addition, for localization, as due to the limited FOV, there is a limitation to detect the landmarks. Therefore, it is also assumed that the landmarks are within the image plane throughout the task.

Due to the indoor localization constraints, Optitrack system has a limited workspace. Hence, in this method, we use an single-view distance estimation approach to enhance the workspace for motion task. A result of the method, we design a localization for the leader agent as well as follower agents for keeping distances between each other during the formation mission. Therefore, in order to setup the method in implementation we use a switching from Optitrack localization to our method when the method is feasible in terms of FOV. Also, it is assumed that agents work in detectable range of landmarks due to FOV.

Chapter 6

Conclusion and Future Works

In this thesis, we have designed a two-level, hierarchical, distributed control scheme to maintain a rigid autonomous formation for a multi-quadrotor system, focusing on real-time implementations. The performance of the developed control scheme has been analyzed by simulations and experimental results for various scenarios. In both simulation and experimental tests, it is clearly seen that our proposed control algorithms achieve desired formation specifications and perform well in all the scenarios. Later, in order to suppress the effects of parametric inertial uncertainties, we have designed an on-line estimator in the pitch and roll dynamics, and developed and adaptive version of our distributed formation control scheme.For this, an adaptive linear quadratic control scheme is designed based on the on-line parameter estimation used with a lookup gain table. The performance of adaptive formation control algorithm has been evaluated via simulation and experimental tests. Successful formation results show the efficiency of the proposed on-line identification and adaptive control scheme. As the performance measure we use the mean square of distance errors in formation of the quadrotors. Both in the real-time experiments and simulations based realistic models of the quadrotors (developed from first principle dynamics as well as using data collected from real-time experiments), we witness the high performance of our adaptive control design in terms of the mean square error. Finally, in order to enhance the current localization system, we have developed a self-localization method for the quadrotor agents based on single-view vision information. This method may be applied to determine global positions of the agents more accurately as well as to estimate distances between agents to satisfy the formation maintenance problem of the multi-quadrotor system.

As a future study, it would be a good extension to combine the single-view distance-estimation based localization enhancement algorithm with the proposed formation control algorithms, and experimentally validate the new control scheme. This way would enable us to control formations employing less sensors and also would increase the workspace of the quadtortors. Two further future directions are analysis of system robustness to actuator failures, and study of other cooperative control tasks such as cooperative surveillance and joint tasks with ground vehicle networks.

References

[1] S. Guler, N. Koksal, and B. Fidan. Adaptive control of a three-agent surveillance swarm with constant velocity constraint. In *Proc. Asian Control Conference*, June 2013.

[2] A. Sutton, B. Fidan, and D. van der Walle. Hierarchical uav formation control for cooperative surveillance. In *Proc. 17th World Congress of International Federation of Automatic Control (IFAC'08)*, pp. 12087-12092, Seoul, Korea, July 2008.

[3] I. Bayezit and B. Fidan. Distributed cohesive motion control of flight vehicle formations. *IEEE Trans. on Industrial Electronics*, vol. 60, no. 12, pp. 5763-5772, December 2013.

[4] B. Fidan, V. Gazi, S. Zhai, N. Cen, and E. Karatas. Single-view distance-estimation-based formation control of robotic swarms. *IEEE Trans. on Industrial Electronics*, vol. 60, no. 12, pp. 73-82, February 2004.

[5] J.P. Desai, J.P. Ostrowski, and V. Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE Trans. Robotics and Automation*, vol. 17, no. 6, pp. 905-908, December 2001.

[6] W. Ren and R.W. Beard. Decentralized scheme for spacecraft formation flying via the virtual structure approach. *Journal of Guidance,Control, and Dynamics*, vol. 27, no. 1, pp. 5781-5791, December 2013.

[7] D.B. Edwards, T.A. Bean, D.L. Odell, and M.J Anderson. A leader-follower algorithm for multiple auv formations. In *Autonomous Underwater Vehicles, 2004 IEEE/OES*, pp. 40-46, June 2004.

[8] J.A. Fax and R.M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Trans. on Automatic Control*, vol. 49, no. 9, pp. 1465-1475, September 2004.

[9] T. Balch and R.C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Trans. on Robotics and Automation*, vol. 14, no. 6, pp. 926-939, December 1998.

[10] A.K. Das, R. Fierro, V. Kumar, J.P. Ostrowski, J. Spletzer, and C.J. Taylor. A vision-based formation control framework. *IEEE Trans. on Robotics and Automation*, vol. 18, no. 15, pp. 813-825, October 2002.

[11] S.G. Loizou and V. Kumar. Biologically inspired bearing-only navigation and tracking. In *Proc. 46th IEEE Conference on Decision and Control*, pp. 1386-1391, New Orleans, LA, USA, December 2007.

[12] S Duran and V. Gazi. Adaptive formation control and target tracking in a class of multi-agent systems. In *Proc. American Control Conference*, pp. 75-80, December 2013.

[13] E. Gul and V. Gazi. Adaptive internal model based formation control of a class of multi-agent systems with switched exosystems. In *Proc. American Control Conference*, pp. 4800-4805, July 2010.

[14] H.G. Tanner, G.J. Pappas, and V. Kumar. Leader-to-formation stability. *IEEE Trans. on Robotics and Automation*, vol. 20, no. 3, pp. 443-455, June 2004.

[15] F. Giulietti, L. Pollini, and M. Innocent. Autonomous formation flight. *IEEE Control Systems Magazine*, vol. 20, no. 6, pp. 34-44, December 2000.

[16] C. Yu, J.M. Hendrickx, B. Fidan, B.D.O Anderson, and V.D. Blondel. Three and higher dimensional autonomous formations:rigidity,persistence andstructural persistence. *Automatica*, vol. 43, no. 3, pp. 387-402, August 2006.

[17] S. Guler, N. Koksal, B. Fidan, and V. Gazi. Indirect adaptive formation control with nonlinear dynamics and parametric uncertainty. In *Proc. Asian Control Conference*, June 2013.

[18] P. A. Ioannou and B. Fidan. *Adaptive Control Tutorial.* SIAM Society for Industrial & Applied Mathematics, TJ217.I628, ISBN 0-89871-615-2, 2009.

[19] S. Bouabdallah. *Design and control of quadrotors with application to autonomous flying.* Ph.D. dissertation, Ecole Polytech Federale de Lausanne,Lausanne,Switzerland, 2007.

[20] G.M. Hoffman, H. Huang, S.L. Waslander, and C.J. Tomlin. Quadrotor helicopter flight dynamics and control theory and experiment. In *Proc. AIAA Guidance, Navigation and Control Conference and Exhibit*, pp. 1-14, Honolulu, Hawaii, April 2008.

[21] P.J. Mckerrow. Modelling the dragan flyer four-rotor helicopter. In *Proc. IEEE International Conference on Robotics and Automation*, pp. 3596-3601, New orleans,LA, April 2004.

[22] T. Li, Y. Zhang, and B. Gordon. Fault tolerant control applied to a quadrotor unmanned helicopter. In *Proceeding of the 7th ASME/IEE international Conference on Mechatronic and Embedded System and Applications*, pp. 1-10, Washington, DC, USA, August 2011.

[23] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proc. 2011 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2520-2525,2011.

[24] I. Sadeghzadeh, A. Mehta, Y. Zhang, and Rabbath C.A. Fault-tolerant trajectory tracking control of a quadrotor helicopter using gain-scheduled pid and model reference adaptive control. In *Proc. Annual Conference of the Prognostics and Health Management Society*, 2011.

[25] Quanser Inc. *Quanser Qball-X4:User Manual.* Document number:888, 2013.

[26] V. Gazi and B. Fidan. Coordination and control of multi-agent dynamic systems: Models and approaches. In E. Sahin W.M. Spears and A.F.T. Winfield, editors, *Swarm Robotics*. Springer, ISBN 0302-9743, 2007.

[27] F. Bullo, J. Cortes, and S. Marinez. *Distributed Control of Robotic Networks; A Mathematical Approach to Motion Coordination Algorithms.* Princeton University Press, ISBN 978-0-691-14195-4, 2009.

[28] H. Bai, M. Arcak, and J. Wen. *Cooperative Control Design; a Systematic, Passivity-based Approach.* Springer, ISBN 978-1-4614-0013-4, 2011.

[29] B.D.O. Anderson, C. Yu, B. Fidan, and D.V.d. Walle. Uav formation control: Theory and application. In V.D. Blondel, S.P. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*. Springer-Velag, 2008.

[30] B.D.O. Anderson, C. Yu, B. Fidan, and J.M. Hendrickx. Rigid graph control architectures for autonomous formations. *IEEE Control Systems Magazine*, vol. 28, no. 6, pp. 48-63, December 2008.

[31] B. Fidan, B.D.O. Anderson, C. Yu, and j.M. Hendrickx. Persistent autonomous formations and cohesive motion control. In P. Ioannou and A. Pitsillides, editors, *Modelling and Control of Complex Systems*. New York: Taylor& Francis, 2007.

[32] S. Sandeep and B. Fidan. Decentralized cohesive motion control of multi-agent formations. In *Proc. Mediterranean Conf. Control Autom.*, pp. 1-6, July 2006.

[33] NaturalPoint, Inc. `http://www.naturalpoint.com/optitrack/products/v100-r2/`. Accessed: February 18, 2014.

[34] J.D. Anderson. *Fundamentals of Aerodynamics.* Chapter 2, McGraw-Hill ,pp.95-181, 2007.

[35] B.L. Stevens and F. Lewis. *Aircraft control and simulation.* Chapter 1,Jhon Wiley & Sons,pp1-59, 2003.

[36] Point Grey Research Inc. `http://www.ptgrey.com/products/fireflymv/fireflymv_usb_firewire_cmos_camera.asp`. Accessed: March 24, 2014.

[37] R. Hartley and A. Zisserman. Multiple view geometry in computer vision. *Cambridge, U.K.: Cambridge Univ. Press*, pp. 153-157,2003.

[38] Fujinon Inc. http://www.fujifilmusa.com/products/optical_devices/security/vari-focal/1-3-manual/yv33x15sa-2/. Accessed: March 25, 2014.