Cooperative Control of Multiple Wheeled Mobile Robots: Normal and Faulty Situations

Mohamed Atef Mohamed Ismail Kamel

A thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy

Concordia University

Montréal, Québec, Canada

July 2016

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mr. Mohamed Atef Mohamed Ismail Kamel**

Entitled: **Cooperative Control of Multiple Wheeled Mobile Robots: Normal and Faulty Situations**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Mechanical and Industrial Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________________ Chair
Dr. P. Valizadah

______________________________ External Examiner
Dr. I. Sharf

______________________________ Examiner, External to Program
Dr. W. P. Zhu

______________________________ Examiner
Dr. B. Gordon

______________________________ Examiner
Dr. I. Stiharu

______________________________ Supervisor
Dr. Y. M. Zhang

Approved by: ______________________________
Dr. A. Dolatabadi, Ph.D. Program Director
Department of Mechanical and Industrial Engineering

July 14, 2016

____
Dr. Amir Asif, Ph.D.,PEng, Dean
Faculty of Engineering and Computer Science

Abstract

Cooperative Control of Multiple Wheeled Mobile Robots: Normal and Faulty Situations

Mohamed Atef Mohamed Ismail Kamel, Ph.D.
Concordia University, 2016

Recently, cooperative control of multiple unmanned vehicles has attracted a great deal of attention from scientific, industrial, and military aspects. Groups of unmanned ground, aerial, or marine vehicles working cooperatively lead to many advantages in a variety of applications such as: surveillance, search and exploration, cooperative reconnaissance, environmental monitoring, and cooperative manipulation, respectively. During mission execution, unmanned systems should travel autonomously between different locations, maintain a pre-defined formation shape, avoid collisions of obstacles and also other team members, and accommodate occurred faults and mitigate their negative effect on mission execution.

The main objectives of this dissertation are to design novel algorithms for single wheeled mobile robots (WMRs) trajectory tracking, cooperative control and obstacle avoidance of WMRs in fault-free situations. In addition, novel algorithms are developed for fault-tolerant cooperative control (FTCC) with integration of fault detection and diagnosis (FDD) scheme.

In normal/fault-free cases, an integrated approach combining input-output feedback linearization and distributed model predictive control (MPC) techniques is designed and implemented on a team of WMRs to accomplish the trajectory tracking as well as the cooperative task. An obstacle avoidance algorithm based on mechanical impedance principle is proposed to avoid potential collisions of surrounding obstacles. Moreover, the proposed control algorithm is implemented to a team of WMRs for pairing with a team of unmanned aerial vehicles (UAVs) for forest monitoring and fire detection applications.

When actuator faults occur in one of the robots, two cases are explicitly considered: **i)** if the faulty robot cannot complete its assigned task due to a severe fault, then the faulty robot has to get out from the formation mission, and an FTCC strategy is designed such that the tasks of the WMRs team are re-assigned to the remaining healthy robots to complete the mission with graceful performance degradation. Two methods are used to investigate this case: the Graph Theory, and formulating the FTCC problem as an optimal assignment problem; and **ii)** if the faulty robot can continue the mission with degraded performance, then the other team members reconfigure the controllers considering the capability of the faulty robot. Thus, the FTCC strategy is designed to re-coordinate the motion of each robot in the team. Within the proposed scheme, an FDD unit using a two-stage Kalman filter (TSKF) to detect and diagnose actuator faults is presented.

In case of using any other nonlinear controller in fault-free case rather than MPC, and in case of severe fault occurrence, another FTCC strategy is presented. First, the new reconfiguration is formulated by an optimal assignment problem where each healthy WMR is assigned to a unique place. Second, the new formation can be reconfigured, while the objective is to minimize the time to achieve the new formation within the constraints of the WMRs' dynamics and collision avoidance. A hybrid approach of control parametrization and time discretization (CPTD) and particle swarm optimization (PSO) is proposed to address this problem. Since PSO cannot solve the continuous control inputs, CPTD is adopted to provide an approximate piecewise linearization of the control inputs. Therefore, PSO can be adopted to find the global optimum solution.

In all cases, formation operation of the robot team is based on a leader-follower approach, whilst the control algorithm is implemented in a distributed manner. The results of the numerical simulations and real experiments demonstrate the effectiveness of the proposed algorithms in various scenarios.

Acknowledgments

First and foremost, I thank ALLAH (God), the creator, for having made everything possible by giving me strength and courage to do this work. We have no knowledge except whatever he has taught us. He is the all-knowing, the all wise.

Though only my name appears on the cover of this dissertation, a great many people have contributed to its production. I owe my gratitude to all those people who have made this dissertation possible and because of whom my graduate experience has been one that I will cherish forever.

My deepest gratitude is to my advisor, Dr. Youmin Zhang. I have been amazingly fortunate to have an advisor who gave me the freedom to explore on my own, and at the same time the guidance to recover when my steps faltered. He taught me how to question thoughts and express ideas. His patience and support helped me overcome many crisis situations and finish this dissertation.

I would like to express the deep appreciation to Dr. Xiang Yu for his vulnerable helps in many aspects of my research. I would say, the success of this thesis would not have been possible without him.

I am also indebted to my colleagues in my research group, for their support, collaboration, friendship, fruitful discussions, and insightful feedback. Particularly, Khaled A. Ghamry, Ali Askari, Xian Wang, and Yiqun Dong. It was pleasure to work with them in this wonderful working environment.

Most importantly, none of this would have been possible without the love and patience of my family. I would like to express my love and gratitude to my mother; although you are thousands of miles away, you were always there whenever I needed you. I am thankful to you for your inspiration and encouragement. Also, I would like to thank my wife, Dina, and my Kids, Mostafa, Ingy, and Ahmed, for their patience and their support, and for the sacrifices they had to make. Their encouragement and support are key ingredients for any achievements I have ever made.

Finally, I would like to thank my country, Egypt, for funding and supporting my research.

Dedication

To the memory of my father, a smart man whom I still miss everyday

Contents

List of Figures

List of Tables

Nomenclature

B	Damping constant (N.s/rad)
C	Constant corresponding to system calibration
C_x	Controllability matrix
E	Applied voltage (V)
F	Virtual repulsive force (N)
F_L	Virtual repulsive force corresponding to the left obstacle (N)
F_R	Virtual repulsive force corresponding to the right obstacle (N)
F_{max}	Maximum value of virtual force (N)
H	Positive integer
I	Inertia (N.s^2/rad)
I_m	DC motor moment of inertia (gm.cm^2)
J	Performance index
K	Elastic constant (N/m)
K_d	Driving direction of the robot
L_a	Armature inductance (mH)
N_c	Control horizon
N_p	Prediction horizon
P	Positive semi-definite weighting matrix
Q	Positive semi-definite weighting matrix
R	Positive semi-definite weighting matrix
R_a	Armature resistance (Ω)
T	DC motor torque (N.m)
T_h	Prediction and control horizons for nonlinear model predictive control

T_s	Sampling time (sec)
Z	Mechanical impedance of the surrounding environment
α_L	Angle between the robot and the left obstacle (rad)
α_R	Distance between the robot and the right obstacle (rad)
$\hat{e}$	Vector of state tracking error
ω	Robot angular velocity (rad/s)
ω_L	Left wheel angular velocity (rad/s)
ω_R	Right wheel angular velocity (rad/s)
ω_c	Correction in the robot angular velocity to avoid the collision (rad/sec)
ϕ	Robot orientation angle (rad)
ϕ_d	Desired formation angle between the leader and the follower (deg)
ξ	Dynamic compensator state
a	Robot linear acceleration (m/s^2)
b	Viscous friction constant (N.m.s)
d	Actual distance between the robot and the nearest obstacle (m)
d_L	Distance between the robot and the left obstacle (m)
d_R	Distance between the robot and the right obstacle (m)
d_{max}	Maximum robot-obstacle distance (m)
d_{min}	Minimum acceptable robot-obstacle distance to avoid the collision (m)
e	Back emf (V)
f_q	Jacobian matrix with respect to vector q
f_u	Jacobian matrix with respect to vector u
h	Distance between the robot longitudinal axis and each wheel (m)
i	Armature current (Amp)
k_e	Back emf constant (V/rad/sec)
k_t	Torque constant (N.m/Amp)
l_d	Desired formation distance between the leader and the follower (m)
m	Number of robot's control inputs
m_d	Number of driving motors' control inputs
m_l	Number of robot's control inputs after input-output linearization

n	Number of robot's states
n_d	Number of driving motors' states
n_l	Number of robot's states after input-output linearization
p_d	Number of driving motors' outputs
q	Vector of generalized coordinates
r	Radius of wheel (m)
s	Positive integer
u	Vector of robot's control inputs
v	Robot linear velocity (m/s)
v_L	Velocity of the robot's left wheel (m/s)
v_R	Velocity of the robot's right wheel (m/s)
v_c	Correction in the robot linear velocity to avoid the collision (m/s)
z	Vector of robot's new states

List of Abbreviations

ASKF	Augmented state Kalman filter
AUVs	Autonomous underwater vehicles
CCD	Charge-coupled devices
CPTD	Control parametrization and time discretization
FDD	Fault detection and diagnosis
FSC	First-state contractive
FTC	Fault-tolerant control
FTCC	Fault-tolerant cooperative control
FTCS	Fault-tolerant control system
GA	Genetic algorithm
LBMPC	Learning-based model predictive control
LMPC	Linear model predictive control
LMRs	Legged mobile robots
LTI	Linear time invariant
LTV	Linear time-varying
MPC	Model predictive control
MUVs	Multiple unmanned vehicles
NMPC	Nonlinear model predictive control
NRS	Network robot systems
PD	Proportional derivative
PSO	Particle swarm optimization
PWM	Pulse width modulation
RHC	Receding horizon control

TSKF	Two-stage Kalman filter
UAVs	unmanned aerial vehicles
UGVs	unmanned ground vehicles
WMRs	wheeled mobile robots

Chapter 1

Introduction

1.1 Overview

From the literal meaning, mobile robots are the robots that can move from one place to another autonomously, that is, without assistance from external human operators. Mobile robots have the special feature of moving around freely within a predefined workspace to achieve their goals. This mobile capability makes them suitable for replacing human beings in civilian applications as well as military applications. According to the environment in which they move, mobile robots can be classified into unmanned aerial vehicles (UAVs), autonomous underwater vehicles (AUVs), and unmanned ground vehicles (UGVs).

UGVs are distinguished in wheeled mobile robots (WMRs) and legged mobile robots (LMRs). WMRs are very popular because they are appropriate for typical applications with relatively low mechanical complexity and energy consumption. LMRs are suitable for tasks in non-standard environments, stairs, heaps of rubble, etc. Mobile robots also include mobile manipulators (wheeled or legged robots equipped with one or more light manipulators to perform various tasks) (Tzafestas, 2014).

A discussion of such a broad universe of possible UGV systems needs some organizing principle. In fact a taxonomy of UGV systems could be based upon any of a number of characteristics of each system, including (Mohammed, 2013): **i)** the specific reasons for choosing a UGV solution for the application (i.e., hazardous environment, size limitation); **ii)** the system's intended operating environment (i.e., indoor environments, outdoors on roads, general cross-country terrain, etc); **iii)** the vehicle's mode of locomotion (i.e., wheels, legs, tracks); and **iv)** how the vehicle's path is determined (i.e., control and navigation techniques employed).

Starting from the late 1960s where the first mobile robot was developed, whose name was Shakey (Nilsson, 1969), and till now, there have been significant researches on WMRs development, which have found important usage in both military and civilian applications (Tzafestas, 2014; Moravec, 1980; Thorpe et al., 1988; Schwartz, 2000; Thrun et al., 2006; Montemerlo et al., 2008; Kamel, 2009). The most important issues in WMRs development are how they can be more intelligent, autonomous, reliable, and safe.

1.2 Motivation

Due to continuous development of advanced mechatronic, computing and communication technologies in the last two decades, it is now possible to find on-board embedded computers which have more computing power. Exchanging information among mobile robots distributed over an area is now possible by means of off-the-shelf ad-hoc wireless network devices. In addition, there are various small size, light weight sensing devices on the market ranging from laser range sensors to color charge-coupled devices (CCD) cameras. As a result, by exploiting current technology, one can build a group of relatively small robots having satisfactory capabilities within a reasonable cost, these robots can interact together in a cooperative manner. This technology is called network robot systems (NRS) (Sanfeliu et al., 2008), or multiple unmanned vehicles (MUVs) (Zhang and Mehrjerdi, 2013). Compared to a single vehicle, the usage of MUVs has many advantages such as:

- **Multi-tasking**: when using a team of robots, the task can be decomposed into several sub-tasks which can be handled at the same time. Therefore, the mission can be achieved much faster than a single robot, results in reduction of the time of mission execution. For example, using a team of UAVs in forest monitoring and fire detection mission can reduce the time of monitoring and information collection significantly compared with single UAV;

- **Fault-tolerance**: In case of using MUVs, if fault occurs in one or more robots in the team, the other robots can accommodate the occurred faults and mitigate their negative effect on mission execution. This increases the system robustness and reliability, especially in dangerous missions;

- **Cost-effectiveness**: Designing a powerful and versatile robot that is capable of handling different tasks sometimes might not be feasible due to robot size and payloads limitations. However, with a group of robots each has simple functions, cost-effective robots can be built without losing the capability of different tasks handling;

- **Flexibility**: The functionality of a group of robots can be easily changed by combining different robots with different capabilities; and

- **Distribution**: Robots can work simultaneously at different positions in the same workspace. For example, during a surveillance task, the target can be monitored from different positions with different types of sensors by a group of robots. This will provide more detailed and accurate information about the target.

Due to these advantages, robotic networks are applied in many applications in both military and civil applications such as surveillance (Acevedo et al., 2014; Kingston et al., 2008), search and exploration (Nieto-Granda et al., 2014; Hu et al., 2013), cooperative reconnaissance (Balch and Arkin, 1998), environmental monitoring (Marques et al., 2005; Dunbabin and Marques, 2012), and cooperative manipulation (Tanner et al., 2003; Prasad et al., 2015). During mission execution, vehicles are required to travel autonomously between different locations, to avoid collision of obstacles and other team members, and to accommodate faults in individual members.

Within these applications, cooperative/formation control arises because a group of robots can accomplish a mission more effectively by maintaining a pre-defined formation shape. Formation control was inspired by the emergent self-organization observed in nature, like birds flocking and fish schooling (Xie, 2007). Formation control allows for intelligent leaders and single agent planning, while followers can focus on other tasks. Leader-following is a common approach to build formations of MUVs. Formation control has been studied extensively in the literature, with application to the WMRs (Consolini et al., 2008), UAVs (Wang and Xin, 2013), and AUVs (Panagou and Kyriakopoulos, 2013), respectively. The challenge here lies in designing a formation controller that is computationally simple, robust, fault-tolerant, and can be implemented in real time.

In addition to maintaining a formation shape during task executions under normal conditions, it would be great that robots possess a fault-tolerance capability in the presence of faults. Thus, the healthy robots can react correspondingly to eliminate the negative effect on mission completion. Otherwise, the formation shape will be broken while mission completion becomes impossible. Such an objective can be achieved by so-called fault-tolerant cooperative control (FTCC) strategies. Accordingly, adding FTCC algorithm to a team of WMRs become very important from the point of view of robots' safety, reliability, and mission completion. The challenges here are **i)** how to detect the faults, and estimate their value and degree of severity; **ii)** how to take the decision to compensate the fault effect on the mission completion; and **iii)** how to execute this decision.

WMR control is challenging since robot model is nonlinear, multi-variable and nonholonomic, also the basic limitation of WMR control comes from their kinematics. The control of WMR requires the ability of the controller to overcome robot nonlinearities as well as the nonholonomic constraints, so that the robot can be stabilized with sound robustness. During the past decades, many researchers work in those challenges. Though numerous control algorithms are found in the literature, controller design is still challenging. Many control algorithms have been developed to solve the WMR control problem such as a Lyapunov-based nonlinear controllers, adaptive controller, fuzzy controllers, and dynamic feedback linearization.

Recently, WMR control problem is formulated as an optimal control problem, where optimization-based techniques can be applied. One of these approaches is the model predictive control (MPC). In the last decade, MPC has gained more attention in the field of WMR control. Its ability to handle constraints makes it a promising approach for single WMR control and also cooperative control of a team of WMRs. However, the main shortcoming of MPC is its high computational requirement, especially with increasing the number of robots, resulting in the limitation of applying MPC with a team of WMRs in real-time applications. For this reason, the main challenge of applying MPC lies in solving the computational effort problem.

1.3 Thesis Objective

The objective of this thesis can be stated as: controlling multiple WMRs while accomplishing different cooperative missions in both normal case in which no fault occurs for any robot in formation, and faulty case where one or more robots subject to faults, then the other team members can eliminate the fault effect and complete the mission.

1.4 Research Contributions

The main contributions of this dissertation can be summarized as follows:

- Develop a novel algorithm for solving the trajectory tracking problem of a differentially-driven WMR based on a combination of an input-output feedback linearization and linear model predictive control (LMPC). The linear model of the robot with nonlinear dynamics is found through feedback linearization, while LMPC is applied to the linear model. With this approach, the computational effort problem associated with MPC can be avoided;

- Theoretical proof of stability, robustness and convergence of the proposed control algorithm;

- Apply the proposed control algorithm to solve the problem of cooperative control of WMRs in a distributed manner;

- Develop an obstacle avoidance algorithm based on the mechanical impedance concept, and implement it to a robot control system. In case of multiple WMRs, the obstacle avoidance algorithm is also embedded to each robot control system to allow the individual robots to avoid obstacles;

- Real-time implementation of the proposed control algorithm in case of single robot trajectory tracking, and cooperative control of a team of WMRs in both obstacle-free and cluttered environments;

- Use the proposed cooperative control algorithm for a team of WMRs, which is paired with a team of UAVs for forest monitoring and fire detection mission, to solve the problems of UAVs' limited flight time and limited payload;

- Exploit the two-stage Kalman filter (TSKF) for fault detection and diagnosis (FDD) purpose. The advantage of using the TSKF is to simultaneously estimate the states and fault parameters necessary for FTCC design and implementation;

- Design FTCC algorithms capable of reconfigure the formation shape of the team once a severe fault has occurred to one or more robots in the team in the situation where the faulty robot(s) are unable to complete the mission. These algorithms are based on the Graph Theory, optimal assignment, and particle swarm optimization (PSO);

- Enhance the FTCC algorithms to be able to drive the team to complete the mission but with degraded performance if one or more robots subject to faults but can continue the mission. Therefore, the other team members will reconfigure their controllers considering the capability of the faulty robot; and

- Real-time implementation of the proposed FTCC algorithms to a team of WMRs in both regular and severe fault cases.

1.5 Thesis Outline

This thesis is presented in six chapters, after this introductory chapter, the structure of the upcoming chapters reflect the composition of the thesis.

Chapter 2 gives a brief introduction of the nonholonomic mobile robots, the kinematic model of the differentially-driven WMRs, its controllability to stabilize at a given posture and about feasible trajectories. Then, a full review of single WMR control is presented. In addition, a brief introduction on cooperative control classes and a literature review of the existing work on cooperative control and FTCC of WMRs is presented. MPC types, advantages and disadvantages, and a review on WMRs control based on MPC are illustrated. Finally, the experimental setup used for the on-board implementation of the proposed algorithms is presented, with the full description of the sensors, the ground station, and the WMRs under study.

In Chapter 3, single robot trajectory tracking as well as cooperative control of a team of WMRs based on a combination of input-output feedback linearization and LMPC is illustrated. Then, an obstacle avoidance algorithm based on the mechanical impedance principle is presented. Next, pairing of a team of WMRs with a team of UAVs for forest monitoring and fire detection mission is investigated. Finally, simulation and experimental results are presented.

Chapter 4 introduces FTCC algorithm of a team of WMRs. In case of severe actuator fault occurrence, the mission is re-assigned to the remaining healthy robots based on two strategies: the Graph Theory, and the optimal assignment. A motion re-coordination algorithm in case of non-severe fault occurrence is also illustrated. TSKF is proposed for FDD purposes. Simulation results, as well as real-time experiments are presented.

Chapter 5 presents the FTCC algorithm based on the time optimal reconfiguration. A hybrid approach of control parametrization and time discretization (CPTD) and PSO is proposed in this chapter. Moreover, simulation and experimental results are also presented.

Finally, in Chapter 6, conclusions and recommendations for future work are outlined.

Chapter 2

Background, Literature Review, and Preliminaries

This chapter reviews some of the most relevant cooperative control of multiple WMRs in normal and faulty situations that can be found in the literature. Some terms and expressions frequently used throughout this thesis are also defined.

This chapter is organized as follows. First, an overview of the differentially-driven WMR is presented. Its kinematic model, controllability to stabilize at a given posture and about feasible trajectories, and a review of the existing control strategies are presented. Second, a brief review on MPC, its concept, types, advantages and disadvantages, and the existing work of controlling WMRs based on MPC is highlighted. Next, a review for cooperative control of WMRs, their classes and strategies in normal case are presented. In addition, a brief review of applying MPC to a team of multiple WMRs is highlighted. Finally, a review of the existing work on cooperative control of WMRs in faulty situations is presented, in which FDD and FTCC are reviewed.

2.1 Basic Motion Tasks of a Single Wheeled Mobile Robot (WMR)

Two basic approaches are considered for a WMR in an obstacle-free environment:

- **Point-to-point motion (or posture stabilization)**: where the desired goal is to stabilize the robot to a final posture starting from a given initial one; and

- **Trajectory tracking**: in which a specific point on the robot must follow a time-varying trajectory in the Cartesian space.

In WMRs, trajectory tracking control is easier to achieve than posture stabilization (Klančar and Škrjanc, 2007). This comes from the assumption that the wheels are in perfect contact with the ground, resulting in the nonholonomic constraints. Also, according to Blazic (2014), trajectory tracking problem is more important since the nonholonomic constraints and other control goals (such as obstacle avoidance, minimum fuel consumption, and minimum travel time) are implicitly included in the path-planning procedure. Furthermore, the trajectory tracking problem can be extended to more complex schemes such as the cooperative control of multiple WMRs.

2.2 Nonholonomic WMRs

A nonholonomic mobile robot is the one that cannot move in the lateral direction, and can move only in the direction perpendicular to the axis of the driving wheels. Most of WMRs can be considered as nonholonomic mobile robots. As mentioned in Chapter 1, nonholonomic mobile robots have the ability to work in large application domains such as search and exploration, surveillance, transportation and military targets tracking. Due to this wide range of applications, the research of nonholonomic mobile robots has many directions. As the dissertation objective stated in Section 1.4, only the trajectory tracking, and cooperative control of a team of nonholonomic mobile robots are considered.

According to Tzafestas (2014), a nonholonomic constraint (relation) is defined to be a constraint that contains time derivatives of generalized coordinates (variables) of a system and is not integrable. To understand what this means, a *holonomic constraint* can be defined as any constraint that can be expressed in the form:

$$F(q,t) = 0 \tag{2.1}$$

where q is the vector of generalized coordinates, $q = [q_1, q_2, \ldots, q_n]^T$. Then, suppose another constraint with following form:

$$f(q, \dot{q}, t) = 0 \tag{2.2}$$

If this constraint can be converted to the form:

$$F(q, t) = 0, \tag{2.3}$$

then, this constraint is integrable. Therefore, although f in (2.2) contains the time derivatives of q, it can be expressed in the holonomic form (2.3), and it is a holonomic constraint.

Definition 2.1 (Nonholonomic constraint)**.** *A constraint of the form* (2.2) *is said to be nonholonomic if it cannot be expressed in the form* (2.3) *such that to involve only the generalized variables themselves*

2.3 Differentially-Driven WMR

An accurate mathematical model of vehicle behavior is very important for the design of the robot controller. In autonomous mobile robots usually two kinds of modeling are used, kinematic and dynamic. Kinematic modeling does not include mass, torque, inertia, etc. It treats the robot as a point object. Dynamic modeling includes the mass, inertia, slippage, etc. Figure 2.1 shows the schematic view of a differentially-driven WMR.

2.3.1 Kinematic Model of a Differentially-Driven WMR

In a two-wheeled differentially-driven WMR, the wheels of the vehicle are controlled independent of each other, so the rear wheels are active and independent in performing driving and steering of the robot, while the front wheel was added only for stability.

As shown in Figure 2.1, point q is the current posture with coordinates (x, y) and orientation angle ϕ. Assuming non-deforming wheels and robot moves without slipping, then defining the vector $q(t)$ and its derivative as:

$$q = [x \;\; y \;\; \phi]^T, \qquad \dot{q} = \left[\dot{x} \;\; \dot{y} \;\; \dot{\phi}\right]^T$$

where $q(t)$ and $\dot{q}(t) \in \mathbb{R}^n$. Then, the nonlinear kinematic equation of the robot is:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\phi & 0 \\ \sin\phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = S(q)u \tag{2.4}$$

where the matrix $S(q) \in \mathbb{R}^n \times \mathbb{R}^m$, and the control inputs vector $u \in \mathbb{R}^m$. v and ω are the linear and angular velocities, respectively. m and n describe the number of robots' control inputs and states, respectively.

The velocities of the right and left wheels of the robot v_R and v_L, respectively can be presented as:

$$\begin{aligned} v_R &= v + h\omega, \\ v_L &= v - h\omega \end{aligned} \tag{2.5}$$

where h is the distance between the robot longitudinal axis and each wheel. Consequently, the angular velocity of each wheel ω_R and ω_L can be calculated as follows:

$$\begin{aligned} \omega_R &= \frac{v_R}{r}, \\ \omega_L &= \frac{v_L}{r} \end{aligned} \tag{2.6}$$

where r is the radius of wheels.

From equation (2.4), one can obtain

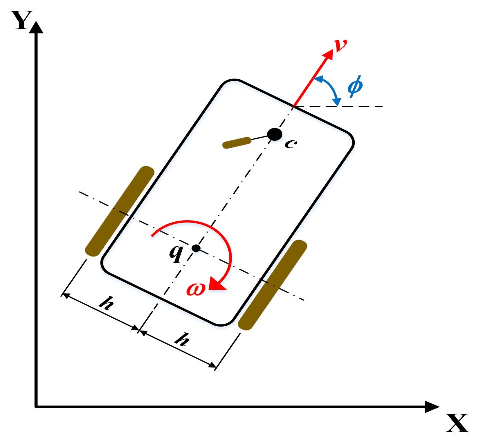


Figure 2.1: Two-wheeled differentially-driven WMR

$$\dot{x} = v\cos\phi,$$

$$\dot{y} = v\sin\phi$$

then

$$\frac{\dot{x}}{\cos\phi} = \frac{\dot{y}}{\sin\phi}$$

which results in

$$\dot{x}\cos\phi = \dot{y}\sin\phi \tag{2.7}$$

Equation (2.7) represent the nonholonomic constraint of the differentially-driven WMR, while Figure 2.2 illustrates this constraint graphically.

2.3.2 Controllability and Stabilization at a Point

The point stabilization can be defined as follows:

Definition 2.2 (Pose stabilization)**.** *Given an arbitrary constant reference position and orientation $q_r = [x_r, y_r, \phi_r]^T$, the point stabilization problem is to find a feedback control law $u = [v, \omega]^T$, such that*

$$\lim_{t\to\infty}(q_r - q_o) = 0,$$

with an initial posture $q_o(0)$.

The first step for analysis and control of nonlinear systems is the linearization of that

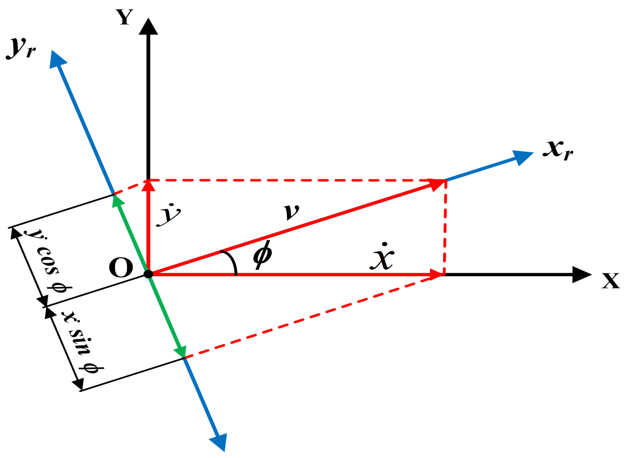


Figure 2.2: Nonholonomic constraint of a differentially-driven WMR

system. If the linear system is controllable, then the original nonlinear system is least locally controllable and feedback stabilizable Xie (2007). Linearizing equation (2.4) about the equilibrium point (where $q = 0$ and $u = 0$), then

$$\dot{q} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2.8}$$

A linear system is said to be controllable at time t_0 if it is possible by means of an unconstrained control vector to transfer the system from any initial state $x(t_0)$ to any other state in a finite interval of time (Ogata, 2010). Mathematically the system will be controllable if and only if the $n \times n$ matrix C_x has full rank, the matrix C_x is given by:

$$C_x = \begin{bmatrix} B & AB & A^2B \dots A^{n-1}B \end{bmatrix} \tag{2.9}$$

If the rank of C_x is n, the linear system is controllable. In other words, it is possible to achieve any point in the state-space of the system by using bounded inputs. By computing the controllability matrix of the linearized model, one can obtain:

$$C_x = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Since the rank of $C_x = 2$, then the linear system is non-controllable.

The model presented in (2.4) belongs to the special class of nonlinear systems, called *affine systems*, and described by the following form:

$$\dot{q} = g_0(q) + \sum_{i=1}^{m} g_i(q)u_i \tag{2.10}$$

The term $g_0(q)$ is called "*drift*", and the system with $g_0(q) = 0$ is called a "*driftless*" system. For driftless affine systems, a sufficient condition for controllability (which is called *the accessibility rank condition*) is that; "for any q, the dimension of the involutive closure of the distribution generated by the vector fields' g_i is equal to n" (De Wit et al., 1993; Yun and Yamamoto, 1992), that:

$$\dim\{\mathrm{inv}\,\Delta\} = n, \qquad \Delta = \mathrm{span}\{g_i\} \tag{2.11}$$

From (2.4), $n = 3$, and the vector fields are:

$$g_1 = \begin{bmatrix} \cos\phi \\ \sin\phi \\ 0 \end{bmatrix}, \quad g_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.12}$$

To determine the involutivity of Δ, we should find the Lie bracket of g_1 and g_2 as

$$g_3 = [g_1, g_2] = \frac{\partial g_2}{\partial q} g_1 - \frac{\partial g_1}{\partial q} g_2 \tag{2.13}$$

then:

$$g_3 = \begin{bmatrix} \sin\phi \\ -\cos\phi \\ 0 \end{bmatrix}$$

so

$$\mathrm{inv}\,\Delta = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ \sin\phi & 0 & -\cos\phi \\ 0 & 1 & 0 \end{bmatrix}$$

It is clear that the rank of $(\mathrm{inv}\,\Delta) = n$. Therefore, the system is controllable. However, the existence of smooth time-invariant state feedback control laws for such nonholonomic systems cannot be implied from controllability. This problem will be discussed in Section 2.3.3.

2.3.3 Brockett's Theorem

The smooth time-invariant state feedback stabilization problem can be defined as:

Definition 2.3. *Find a state feedback $u = k(q)$, where $k(q)$ is a smooth function of q, such that the closed-loop system.*

$$\dot{q} = S(q)k(q) \tag{2.14}$$

is asymptotically stable.

Brockett (1983) gives a general theorem on necessary conditions for smooth feedback stabilization of such nonlinear systems

Theorem 2.1 (Brockett's Theorem)**.** *Consider the nonlinear system $\dot{x} = f(x, u)$ with $f(x_0, 0) = 0$ and $f(.,.)$ continuously differentiable in a neighborhood of $(x_0, 0)$, necessary conditions for the existence of a continuously differentiable control law for asymptotically stabilizing $(x_0, 0)$ are:*

1. *The linearized system has no uncontrollable modes associated with eigenvalues with positive real part,*

2. *There exists a neighborhood N of $(x_0, 0)$ such that for each $\xi \in N$ there exists a control $u_\xi(t)$ defined for all $t > 0$ that drives the solution of $\dot{x} = f(x, u_\xi)$ from the point $x = \xi$ at $t = 0$ to $x = x_0$ at $t = \infty$,*

3. *The mapping $\gamma : N \times \mathbb{R}^m \to \mathbb{R}^n$, N a neighborhood of the origin, defined by $\gamma : (x, u) \to f(x, u)$ should be onto an open set of the origin.*

The details and proof of Brockett's theorem are mentioned in (Brockett, 1983; Bloch, 2003). The following corollary to Brockett's theorem is a special case for driftless systems.

Corollary 2.1. *Consider a driftless system of the form*

$$\dot{q} = \sum_{i=1}^{m} g_i(q) u_i, \quad q \in \mathbb{R}^n, u \in \mathbb{R}^m, m \leq n \tag{2.15}$$

where g_i are smooth vector fields. If the vectors g_i are linearly independent, i.e.

$$\text{rank}[g_1, g_2, \ldots, g_m] = m \tag{2.16}$$

*then a solution to a stabilization problem defined in **Definition 2.3** exists if and only if $m = n$.*

According to **Corollary 2.1**, Lyapunov stability is achieved only if the number of inputs equal to the number of states. Since $n = 3$, and $m = 2$, then a smooth time-invariant feedback laws is not valid for the nonholonomic WMR. Consequently, to get a posture-stabilizing controller, it is either necessary to avoid the continuity requirement or to use time-varying control laws.

2.3.4 Controllability and Stabilization about a Trajectory

In a trajectory tracking problem, it is convenient to generate a desired trajectory with coordinates $q_r = [x_r(t), y_r(t), \phi_r(t)]^T$. In order to be feasible, the reference trajectory must satisfy the nonholonomic constraint on the robot motion, or be consistent with equation (2.4) (Luca et al., 2001). The reference inputs $u_r = [v_r(t), \omega_r(t)]^T$ can be calculated as follows:

$$v_r(t) = \pm\sqrt{\dot{x}_r^2(t) + \dot{y}_r^2(t)} \tag{2.17}$$

$$\omega_r(t) = \frac{\dot{x}_r(t)\ddot{y}_r(t) - \dot{y}_r(t)\ddot{x}_r(t)}{\dot{x}_r^2(t) + \dot{y}_r^2(t)} \tag{2.18}$$

A linearized model is obtained by computing an error model with respect to a reference trajectory. Re-write the nonlinear model (2.4) and the reference trajectory in the general form:

$$\dot{q} = f(q, u) \tag{2.19}$$

$$\dot{q}_r = f(q_r, u_r) \tag{2.20}$$

By expanding the robot model (2.19) in Taylor series around the reference trajectory (q_r, u_r) and discarding the high order terms, then

$$\dot{q} = f(q_r, u_r) + \left.\frac{\partial f(q,u)}{\partial q}\right|_{\substack{q = q_r \\ u = u_r}} (q - q_r) + \left.\frac{\partial f(q,u)}{\partial u}\right|_{\substack{q = q_r \\ u = u_r}} (u - u_r) \tag{2.21}$$

or

$$\dot{q} = f(q_r, u_r) + f_q(q - q_r) + f_u(u - u_r) \tag{2.22}$$

where f_q and f_u are the Jacobian matrices of f with respect to q and u respectively, evaluated around the reference trajectory (q_r, u_r). Subtracting (2.20) from (2.22) will get

$$\dot{q}_e = f_q q_e + f_u u_e \tag{2.23}$$

where $q_e = q - q_r$ and $u_e = u - u_r$, and the Jacobian matrices f_q and f_u can be obtained as:

$$f_q = \begin{bmatrix} 0 & 0 & -v_r \sin\phi_r \\ 0 & 0 & -v_r \cos\phi_r \\ 0 & 0 & 0 \end{bmatrix}, \quad f_u = \begin{bmatrix} \cos\phi_r & 0 \\ \sin\phi_r & 0 \\ 0 & 1 \end{bmatrix}$$

Then, equation (2.23) can be written as:

$$\dot{q_e} = \begin{bmatrix} 0 & 0 & -v_r \sin\phi_r \\ 0 & 0 & -v_r \cos\phi_r \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x - x_r \\ y - y_r \\ \phi - \phi_r \end{bmatrix} + \begin{bmatrix} \cos\phi_r & 0 \\ \sin\phi_r & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v - v_r \\ \omega - \omega_r \end{bmatrix} = A(t)q_e + B(t)u_e \qquad (2.24)$$

Since the linearized system is time varying, then a necessary and sufficient controllability condition is that the controllability Gramian is nonsingular (Luca et al., 2001). A simple analysis can start with determining the state tracking error $\hat{e} = [x_e(t), y_e(t), \phi_e(t)]^T$ as shown in Figure 2.3, such that:

$$\hat{e} = \begin{bmatrix} x_e \\ y_e \\ \phi_e \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_r \\ y - y_r \\ \phi - \phi_r \end{bmatrix} \qquad (2.25)$$

The associated tracking error is obtained by differentiating equation (2.25), then:

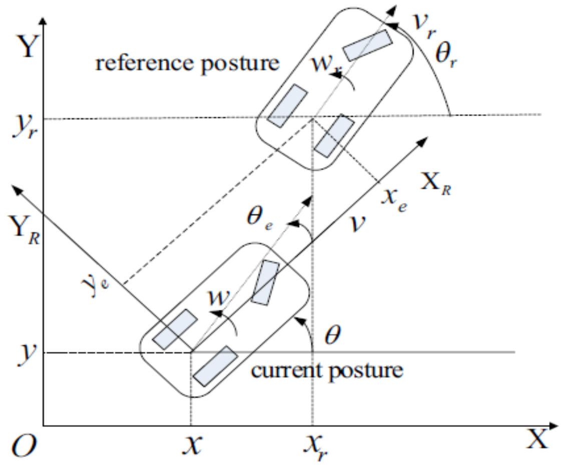


Figure 2.3: Trajectory tracking problem

$$\dot{\hat{e}} = \begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\phi}_e \end{bmatrix} = \begin{bmatrix} \omega y_e - v + v_r \cos\phi_e \\ -\omega x_e + v_r \sin\phi_e \\ \omega_r - \omega \end{bmatrix} \tag{2.26}$$

Introducing the following new inputs:

$$\begin{aligned} u_1 &= -v + v_r \cos\phi_e \\ u_2 &= \omega_r - \omega \end{aligned} \tag{2.27}$$

then equation (2.26) can be represented as:

$$\dot{\hat{e}} = \begin{bmatrix} 0 & \omega & 0 \\ -\omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ \phi_e \end{bmatrix} + \begin{bmatrix} 0 \\ \sin\phi_e \\ 0 \end{bmatrix} v_r + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{2.28}$$

Linearizing equation (2.28) around the reference trajectory ($\hat{e} = 0$, $u = 0$) will yield the following time-varying system:

$$\dot{\hat{e}} = \begin{bmatrix} 0 & \omega_r(t) & 0 \\ -\omega_r(t) & 0 & v_r(t) \\ 0 & 0 & 0 \end{bmatrix} e + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{2.29}$$

If v_r and ω_r are constant, the system (2.29) becomes time-invariant. Check the controllability of that system by calculating the matrix C_x as mentioned in (2.9), then:

$$C_x = \begin{bmatrix} 1 & 0 & 0 & 0 & -\omega_r^2 & v_r\omega_r \\ 0 & 0 & -\omega_r & v_r & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

In this case, the system (2.29) is controllable either if v_r or ω_r are nonzero, and the rank of the controllability matrix is 3. This implies that smooth stabilization is possible and linear controllers can be used to achieve the stabilization for feasible trajectories, as long as they do not come to stop.

Then, the trajectory tracking problem can be defined as follows:

Definition 2.4 (Trajectory tracking)**.** *The trajectory tracking problem under the assumption that the reference robot is not as rest* ($v_r = \omega_r = 0$) *when* $t \to \infty$ *is to find a feedback control law* $u = [v, \omega]^T$*, such that*

$$\lim_{t\to\infty} (q_r - q_0) = 0,$$

with an initial posture $q_0(0)$*.*

2.3.5 Differentially-Driven WMR Control

In this section, a brief literature review for a trajectory tracking problem of a differentially-driven WMR is presented. The control of WMR requires the ability of the controller to overcome robot nonlinearities as well as the nonholonomic constraint, so that the robot can be stabilized with sound robustness. During the past decades, many researchers work on those challenges. Though numerous control algorithms are found in the literature, the controller design is still challenging. Many control algorithms have been developed to solve the WMR control problem. Lyapunov based method is presented in (Kanayama et al., 1990; Samson, 1993; Blažič, 2011). Dynamic feedback linearization is applied in (d'Andréa Novel et al., 1995; Oriolo et al., 2002; Chwa, 2010). Sliding mode control is presented in (Yang and Kim, 1999; Koubaa et al., 2013). In (Oya et al., 2003) researchers proposed controllers by converting the system into a chained form. MPC is proposed in (Lim et al., 2008; Klančar and Škrjanc, 2007; Kuhne et al., 2004).

An important issue in the controllers mentioned above is that they are designed based on the robot kinematics only. However, in case of high speed movements and/or heavy weights, it becomes necessary to consider robot dynamics in addition to its kinematics (Martins et al., 2008). The common concept of considering both dynamics and kinematics is the backstepping technique. The steps of backstepping control are:

1. Design the velocity controller based on the kinematic system (as mentioned above);

2. Design a feedback velocity-following controller based on robot dynamics that the robot's actual velocities converge to the velocities generated by the first controller; and

3. Calculate the required torques (which will be the control signals) that drive the robot to follow the desired trajectory.

Most of the dynamic controllers found in literature generate torques as a control signals (Fierro and Lewis, 1995; Das and Kar, 2006). The drawback of these controllers is that most of the commercial robots receive velocity commands as control signals. Therefore, in (De La Cruz and Carelli, 2006; Martins et al., 2008; Taheri-Kalani and Khosrowjerdi, 2014) they presented dynamic controllers with velocities considered as control inputs.

The main problem with the controllers based on dynamics and kinematics is their complexity compared to those based on kinematics only. The complexity comes from considering both kinematics and dynamics, and due to some unknown parameters and some parametric uncertainties such as lateral and longitudinal slipping. Some of these parameters are unknown. So, the controller should be enhanced to estimate and identify these unknown parameters. Fuzzy control, neural networks and adaptive control are efficient to estimate these parameters and uncertainties and to solve the trajectory tracking problem.

From a review of the literature, the following results are summarized for the problem of nonholnmomic WMR trajectory tracking and point stabilization:

- The robot model is nonlinear, multi-variable, and nonholonomic;
- A nonholonomic WMR is controllable and its equilibrium point can be made Lyapunov stable, but cannot be made asymptotically stable by a smooth static state feedback (Campion et al., 1991);
- In posture stabilization: it is either necessary to avoid the continuity requirement and/or to use a time-varying control law (Campion et al., 1991);
- Nonholonomic WMRs can track a pre-defined trajectory and smooth stabilization is possible as long as the desired reference linear and angular velocities v_r or ω_r are nonzero (Oriolo et al., 2002);
- It has been shown that nonholonomic WMRs are not input-state linearizable. However, they are input-output linearizable (Yun and Yamamoto, 1992; Shojaei et al., 2013);

- The trajectory tracking problem is easier to solve and more important than posture stabilization (Klančar and Škrjanc, 2007);
- Controller design based on kinematic model is efficient in case of low speeds;
- In case of heavy weights and higher speeds, dynamic model should be considered in controller design; and
- Most of the WMRs use velocities commands as control input signals.

2.4 Model Predictive Control (MPC)

2.4.1 Overview

Recently, MPC, also known as receding horizon control (RHC), received a great attention in the control community, due to its ability to solve multi-variable constrained problems. Although it has been used for a long time in some industrial processes such as oil refinery, biomedical industry, and chemical plants (Pan and Wang, 2012), MPC applied recently with UAVs (Mahony et al., 2012; Abdolhosseini et al., 2013; Hafez et al., 2014) and WMRs (Klančar and Škrjanc, 2007; Lim et al., 2008).

The importance of applying MPC in the control community comes from its ability to handle the states and inputs constraints, and real-time predication, optimizing and correcting the feedback. Compared to the conventional control methods that use pre-computed control laws, MPC family is based on iterative, finite horizon optimization of a plant model to obtain an estimate of its future behavior. An optimization problem based on a performance cost function is then solved to choose an optimal sequence of controls from all feasible sequences. The first control input of this optimal sequence is then applied to the feedback control loop, and the whole procedure is repeated at each subsequent time step. Figure 2.4 shows the basic structure of MPC. The main key principle of building an MPC controller can be summarized as follows (Lazar, 2006):

- Calculate the predictions of the future system behavior based on the explicit use of plant model;

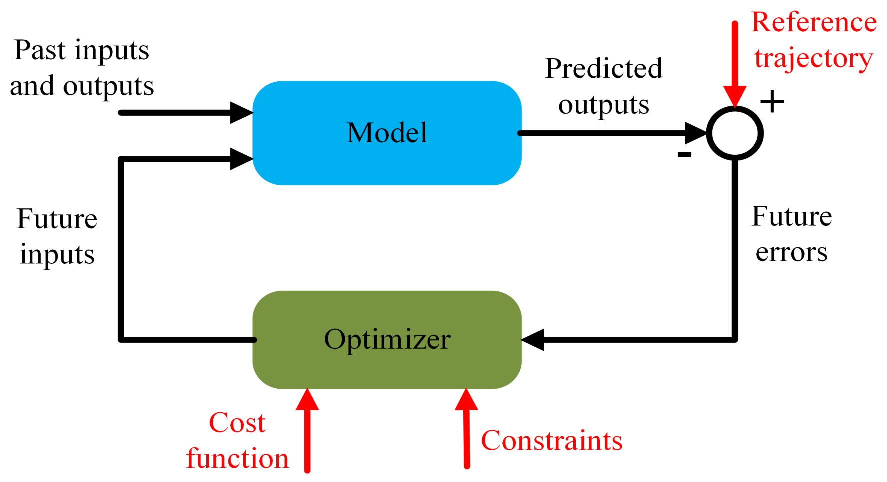


Figure 2.4: Basic structure of MPC (Camacho and Bordons, 2007)

- Optimize the objective function subject to constraints, results in the optimal sequence of controls; and
- Use the receding horizon strategy, in which only the first element of the optimal sequence of controls is applied on-line.

In order to reduce the computational burden, MPC uses both a control horizon and a prediction horizon. The control horizon determines the number of actuation signals to find. On the other hand, the prediction horizon determines how far the behavior of the system is predicted.

The MPC methodology involves solving an on-line open loop finite horizon optimal control problem subject to input, state, and/or output constraints. As shown in Figure 2.5, at a time t, the system model and the measured variables (outputs) are used to predict the future behavior of the controlled plant over the prediction horizon N_p. Usually, the system's future response is expected to return to a desired set point by following a reference trajectory from the current states. The difference between the predicted output and the reference trajectory is called predicted error. A finite horizon optimal control problem with a performance index (usually be minimizing the predicted control input and the predicted error) is solved on-line and an optimal control input $u^*(t)$ over a control horizon N_c (usually $N_c \leq N_p$), which minimizes the predicted error, is obtained. Only the first element of $u^*(t)$ is implemented to the plant. All the other elements are discarded. Then, at the next time

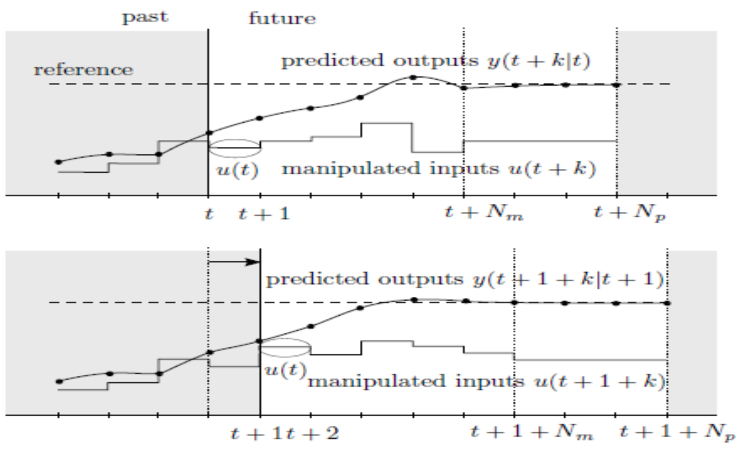

past
future
reference
predicted outputs $y(t+k|t)$
$u(t)$
manipulated inputs $u(t+k)$
t
$t+1$
$t+N_m$
$t+N_p$
predicted outputs $y(t+1+k|t+1)$
$u(t)$
manipulated inputs $u(t+1+k)$
$t+1$
$t+2$
$t+1+N_m$
$t+1+N_p$

Figure 2.5: MPC scheme

interval, the whole procedure is repeated.

The previous methodology can be mathematically formulated as follows: Consider the following discrete-time linear system:

$$x(k+1) = Ax(k) + Bu(k) \tag{2.30}$$

where $x(k) \in \mathbb{R}^n$ and $u(k) \in \mathbb{R}^m$, n is the number of states and m is the number of control inputs. $A \in \mathbb{R}^{n\times n}$, and $B \in \mathbb{R}^{n\times m}$. Then, at each time interval k, MPC can be formulated as the following optimization problem

$$\min_{u(\cdot)} J_{(N_p,N_c)}(x_k) \tag{2.31}$$

subject to

$$\begin{aligned} &x(k+i|k) = Ax(k+i-1|k) + Bu(k+i-1|k), \\ &x(k+i|k) \in \mathcal{X}, \\ &u(k+i|k) \in \mathcal{U} \end{aligned} \tag{2.32}$$

The performance index J can be defined as:

$$\begin{aligned} J_{(N_p,N_c)}(x_k) =& x^T(k+N_p)Px(k+N_p) + \sum_{i=1}^{N_p-1} x^T(k+i|k)Qx(k+i|k) \\ &+ \sum_{i=0}^{N_c} u^T(k+i|k)Ru(k+i|k) \end{aligned} \tag{2.33}$$

where $P \in \mathbb{R}^{n\times n}$, $Q \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{m\times m}$ are the three positive semi-definite weighting matrices with $P > 0$, $Q > 0$ and $R > 0$. The weighting coefficients of P, Q, and R reflecting the relative importance of the final state error cost, the intermediate state error cost, and the control input error cost, respectively. $\mathcal{X} \subset \mathbb{R}^n$ are the state constraints. $\mathcal{U} \subset \mathbb{R}^m$ are the input constraints. Usually, $\mathcal{U} = \{u \in \mathbb{R}^m : u_{min} \leq u \leq u_{max}\}$. u_{min} and u_{max} are known constants in $\mathbb{R}^m$. The first term on the right hand side of equation (2.33) is called the terminal state penalty, the second term is called the state penalty and the last term is called the control penalty.

In nonlinear systems, MPC concept is still the same. Consider the following continuous time nonlinear system:

$$\dot{x}(t) = f(x(t), u(t)) \tag{2.34}$$

where $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$, n is the number of states and m is the number of control inputs. As in the linear case, MPC can be formulated as the following optimization problem (Xie, 2007):

$$\min_{u(\cdot)} J(x(t), u(\cdot)) \tag{2.35}$$

subject to

$$\begin{aligned} &\dot{x}(t) = f(x(t), u(t)), && \\ &x_{min} \leq x(s; x(t), t) \leq x_{max}, && t \leq s \leq t + T_h, \\ &u_{min} \leq u(s) \leq u_{max}, && t \leq s \leq t + T_h \end{aligned} \tag{2.36}$$

The performance index J can be defined as:

$$J_{T_h}(x(t), u(\cdot)) = \int_t^{t+T_h} (\|\bar{x}(s; x(t), t)\|_Q^2 + \|u(s)\|_R^2) ds + \|\bar{x}(t + T_h; x(t), t)\|_P^2 \tag{2.37}$$

where T_h represent both the prediction and the control horizons.

The advantages of MPC can be summarized as (Camacho and Bordons, 2007; Hafez, 2014):

- It can deal with multi-variable and nonlinear systems;
- It is very useful when future references are known;

- Higher efficiency based on the minimization of the cost function;
- It allows operation within constraints; and
- It can handle multiple systems easily by merging them into the objective function;

However, the main disadvantages of MPC are (Hafez, 2014):

- As the system complexity increases, the on-line calculations of the control law become less feasible;
- Its computational time is high especially in real-time applications;
- In case of closed-loop systems, it is difficult to predict the controller performance; and
- Theoretical results regarding stability and robustness are not easily applied to general cases.

2.4.2 WMRs Control Based on MPC

Although MPC is not a new control approach, but a few works deal with control of WMRs by means of MPC are found in the literature. MPC can be classified into nonlinear model predictive control (NMPC) and LMPC.

van Essen and Nijmeijer (2001) presented an NMPC with time varying weights which is applied for both trajectory tracking and posture stabilization, simulation results are presented. Gu and Hu (2004) used a stabilizing NMPC to achieve simultaneous tracking a pre-defined trajectory. Stability is addressed in this work by forcing the terminal state to move into a terminal state region through adding a stability term to the cost function. Simulation results are presented. Seyr and Jakubek (2005) presented an NMPC algorithm considering side slip and tangential wheel slip. Predicted future position errors are minimized by numerical computation of an optimal sequence of control inputs using pre-specified shape functions based on a Gauss-Newton algorithm. Simulation results are presented. Kühne et al. (2005) presented both NMPC and LMPC approaches for trajectory tracking of a differentially-driven WMR. The results show that the computational effort of NMPC is much higher than in the case of LMPC. Xie and Fierro (2008) developed a first-state contractive (FSC) MPC for trajectory tracking and point stabilization. Stability is guaranteed by adding a first-state contractive constraint at the beginning of the prediction horizon. Simulation results are presented. Lim et al. (2008) proposed an NMPC approach to be applied for trajectory tracking and obstacle avoidance of a single WMR. A nonlinear-programming problem is solved on-line, and simulation results were presented. Kanjanawanishkul et al. (2009) also presented an NMPC approach for trajectory tracking of a differentially-driven WMR and experimental results are presented. Ma et al. (2012) presented an NMPC approach based on both dynamics and kinematics. Stability is guaranteed by adding a terminal state penalty to the cost function and constraining the terminal state to a terminal region. The terminal region and its corresponding local controller are developed based on T-S fuzzy model. Simulation results are presented.

As noticed above, applying NMPC in real-time application is very few. In NMPC, a nonlinear programming problem to be solved on-line, which is usually non-convex. As a result, the computational time increases. Therefore, the main demerit of using NMPC is that the computational burden is much higher. Consequently, applying NMPC to a single WMR or a team of robots is limited in real-time applications.

To avoid the computational efforts problem associated with NMPC, LMPC approach is proposed. Kuhne et al. (2004) and Lages and Alves (2006) presented a linear time-varying (LTV) description of the robot model based on linearizing the error dynamics between the reference trajectory and the actual one. The states of the linearized system are the errors in the position in x and y directions and the error in the orientation angle ϕ. The control law is derived by the optimization of a quadratic cost function. Simulation and experimental results are presented. Jiang et al. (2005) presented a combination between LMPC and a fuzzy control. LMPC is used to predict the position and the orientation of the robot and the fuzzy control is used to deal with the nonlinear characteristics of the system. Experimental results are presented. In (Klančar and Škrjanc, 2007), LMPC is applied for single WMR trajectory tracking problem based on linearizing the error dynamics model between the reference trajectory and the actual one. The objective function is to minimize the difference between the future trajectory-following errors of the robot and the reference trajectory. The control law is explicitly obtained without using any optimization solver, while the bounded velocity and acceleration constraints are considered in the low-level controller implemented in the robot. Experimental results showed the effectiveness of the proposed control law compared to a state-tracking controller presented in (Kanayama et al., 1990). In (Maurovic et al., 2011), the authors presented an explicit LMPC scheme, where the solution of the minimization problem can be calculated off-line and expressed as a piecewise affine function of the current state of the robot, thus avoiding the need for on-line minimization. By obtaining such optimal controller, which has a form of a look-up table, there is no need for expensive and large computational infrastructure. Experimental results are presented. However, since all the computations are calculated off-line, then this method cannot guarantee that the robot can continue tracking the desired trajectory in case of sudden situations such as fault occurrence and facing any obstacles.

As noticed above, using LMPC was based on an LTV model, meaning that, at each sample, model states are changed. As a result, applying LMPC is possible for trajectory tracking of WMRs. However, with increasing the number of robots, the computational effort will be a great challenge.

2.5 Cooperative Control of WMRs

Cooperative multi-agent system is composed of agents that can operate together to perform some global task. In this sense, such a control issue is referred as *cooperative control*, which highlights the cooperation function of the agents during operation. Formation control of multiple agents can be considered as a special cooperative operation. The objective of formation is to keep certain shape with constant relative distances between the agents during mission execution. As a result, shape and position are the two important characteristics of a formation configuration. The following definition describes the meaning of formation.

Definition 2.5 (Formation). *A formation is a network of agents interconnected via their controller specifications that dictate the relationship each agent must maintain with respect to its neighbors. The interconnections between agents are modeled as edges in a directed acyclic graph, labeled by a given relationship (Tanner et al., 2004).*

A group of robots can take different formation shapes, the most common shapes are line, triangle, diamond, wedge, etc. Besides the shape, each robot in the team must have a specific position in the formation. There are three major techniques for formation position as mentioned in (Balch and Arkin, 1998), a unit-center-referenced, a leader referenced, and a neighbor-referenced as shown in Figure 2.6.

In the *unit-centered-referenced* position, each robot maintains its own position relative to a center point. This center is the average of the x and y positions of all the robots involved in the formation. In the *leader-reference* position, each robot (except the leader) determines its formation position relative to the lead robot. In the *neighbor-referenced* position, each robot maintain its desired position with respect to one other predetermined robot.

2.5.1 Cooperative Control Strategies

Various approaches and strategies have been proposed for the formation control. The most common approaches are virtual structure, behavior-based, leader-follower, graph-based, and artificial potential approaches (see Zhang and Mehrjerdi, 2013, and the references therein). Figure 2.7 illustrates the block diagram for these strategies.

The virtual structure approach is proposed by Lewis and Tan (1997), and studied in (Beard et al., 2001; Egerstedt and Hu, 2001; Ren and Beard, 2004; Ghommam et al., 2010; Liu and Jia, 2012). The main idea is that the entire formation is treated as a single entity as shown in Figure 2.8, and the desired motion is assigned to the virtual structure that traces trajectories for each member in the formation to follow. The advantages of this approach are the simplicity since it is easy to prescribe the coordinated behavior of the whole team.

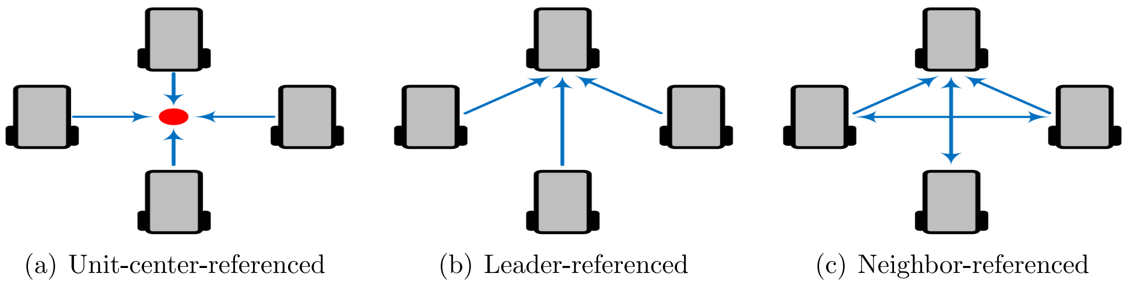

(a) Unit-center-referenced (b) Leader-referenced (c) Neighbor-referenced

Figure 2.6: Formation position determination by different referencing techniques

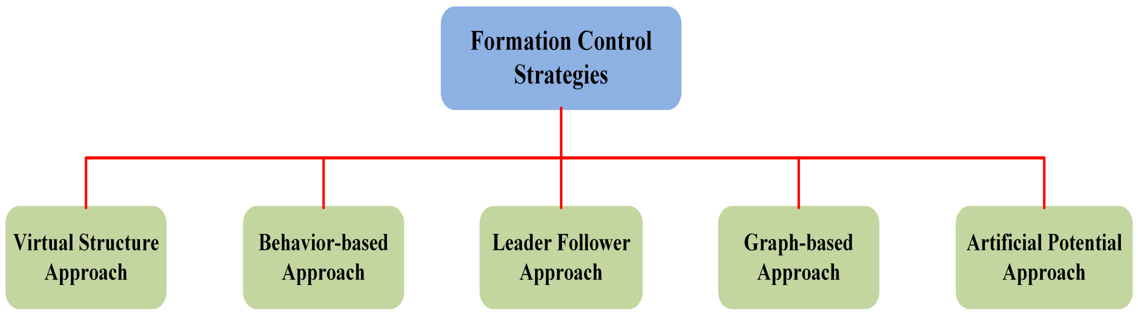

Formation Control Strategies
Virtual Structure Approach
Behavior-based Approach
Leader Follower Approach
Graph-based Approach
Artificial Potential Approach

Figure 2.7: Formation strategies for multiple robots

Moreover, during maneuvers, it is easy to maintain the formation and the rigid geometric relationships among the robots. However, its main disadvantage is the centralization that needs more communication loads.

Inspired by formation behaviors in nature like schooling and flocking, the behavior-based formation control approach is proposed. In this approach, several desired behaviors are prescribed for each vehicle and the final control is derived from a weighting of the relative importance of each behavior. Possible behaviors can be goal seeking, obstacles and collision avoidance, and formation keeping, as shown in Figure 2.9. The advantage of this approach are the decentralization and it can be implemented with less communications. However, it is difficult to do mathematical analysis for robustness and stability. Behavior-based approach is studied in (Sugihara and Suzuki, 1996; Lawton and Beard, 2002; Lawton et al., 2003;

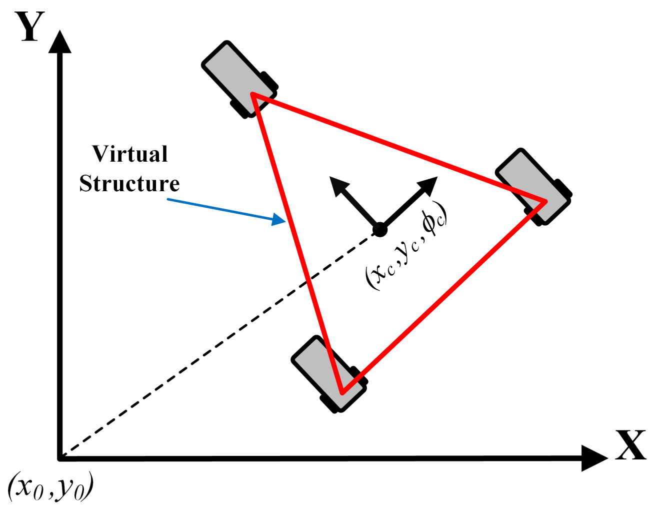


Figure 2.8: Virtual structure approach

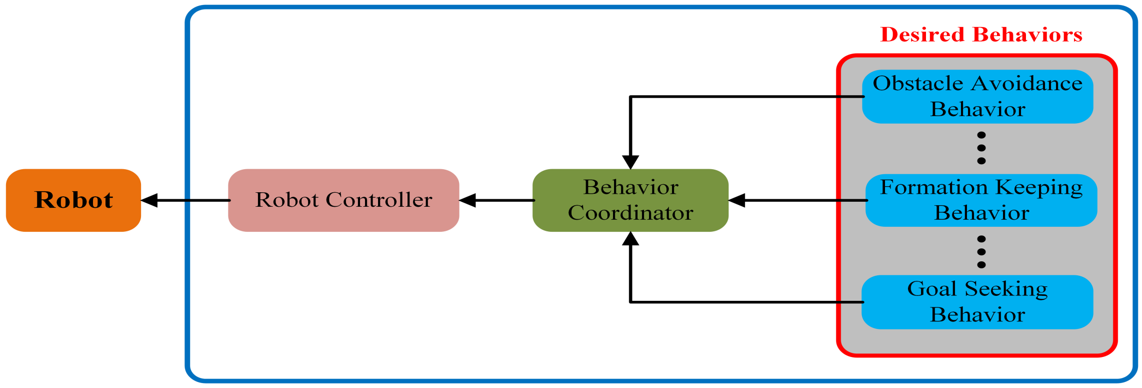


Figure 2.9: Behavior-based approach

Takahashi et al., 2004; Antonelli et al., 2009).

In the leader-follower method (De La Cruz and Carelli, 2006; Sira-Ramírez and Castro-Linares, 2010; Klančar et al., 2011), one of the robots is assigned as a leader, while other robots are followers. During motion, only the leader's motion and the desired relative positions between the leader and the followers are required. Once the motion of leader is given, local control law on each follower can achieve the desired relative position of the follower with respect to the leader. Thereby the desired formation of the entire system is achieved and maintained. Consequently, the formation control problem can be viewed as a natural extension of the traditional single WMR trajectory tracking problem (Li et al., 2004). Based on this fact, many approaches commonly used in trajectory tracking of WMR are applied to design a control law for the leader-follower approach. In addition, a few works present both the path planning and formation control problems together (Garrido et al., 2011; Saska et al., 2014). Figure 2.10 shows a scheme of the leader-follower approach where the follower should follow the leader maintaining the desired formation (l_d - ϕ_d) where l_d is the desired formation distance and ϕ_d represent the desired formation angle.

The graph-based approach was proposed by Desai et al. (1998) and studied in (Fax and Murray, 2004; Jin and Murray, 2004; Dong and Guo, 2007). In this approach, formation can be described by graphs, and the main purpose of these graphs are to give a mathematical representation of the structure of the robots network. Each robot treated as a vertex, and the edges that connect the vertices represent the information flow from one vertex to another. The graph represents the allowed information flow between the agents. Graph-based

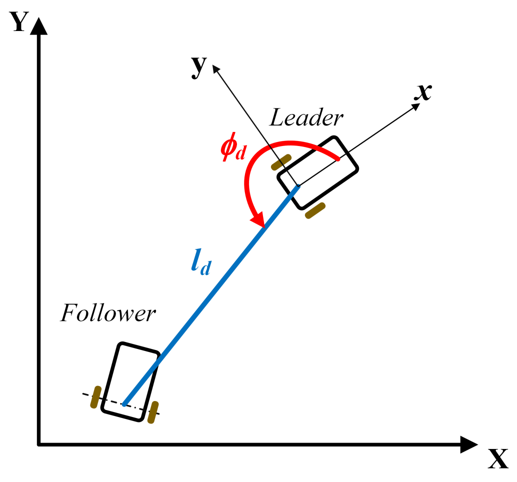


Figure 2.10: Leader-follower approach

approach have many advantages such as decentralization, and the team can keep a suitable behavior even in the presence of varying communication topology. However, the main disadvantage is that agents can only get information from their neighbors.

Formation control based on artificial potential field is studied in (Ogren et al., 2004; Ge and Fua, 2005; Masoud, 2007; Do, 2008). The main idea is to use the artificial potentials to define the interaction control forces between neighboring agents, and designed to enforce the inter-agent spacing. The main advantages of this approach that it can be extended to use in collision avoidance of agents. Nevertheless, it suffers instability due to the delays in the communication channels.

2.5.2 Cooperative Control Classes

The cooperation between groups of robots requires sharing information between team members via a good communication network, allowing them to act as one unit, and execute the assigned mission. Each robot can make its decision. However, having a cooperative decision algorithm allows efficient mission execution, and increases the ability of the team to tolerate any fault may occur to one more members of the team of robots.

Classes of cooperative control are defined as different control techniques used in controlling a group of multiple robots (Hafez, 2014). From this point of view, two main classes are considered; a centralized control and a distributed/decentralized control (Zhang and

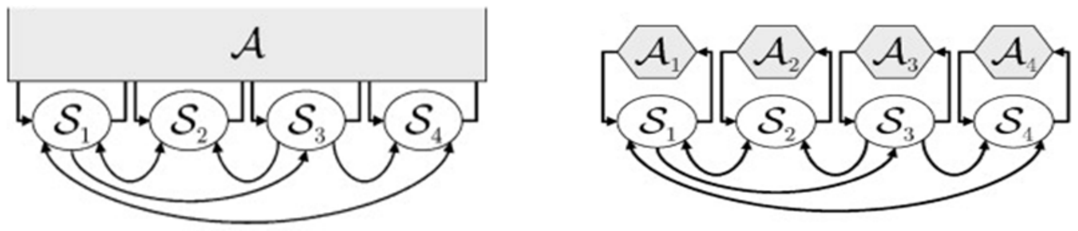
Mehrjerdi, 2013).

In the centralized control (De La Cruz and Carelli, 2006; Mehrjerdi et al., 2011), a group of cooperative robots receives the control commands from a powerful core unit. This unit can be either a robot in the formation with large computing capability, or a ground control station equipped by large computing and communication equipment. The core unit can communicate with the robots in the team. It receives information from all members, optimize vehicle coordination, monitor mission accomplishment, and accommodate individual vehicle faults. Centralized control is characterized by the presence of global information, centralized organization structure, and high computational performance. However, centralized approach is less robust, high cost, and highly sensitive to failure. Moreover, with increasing the number of robots, centralized control become inefficient due to the communication limits. As a result, applying centralized control is limited in real-time missions.

To solve these problems, decentralized control was developed; each robot can communicate and share information with the other team members, and can only achieve its specified task as part of the global mission. Decentralized control characterized by local information and decentralized organization structure. It has many advantages such as scalability, robustness to individual faults, less communication load and computational power of the robots' controllers. However, it lacks the good performance of the centralized approach. Figure 2.11 illustrates the difference between centralized and decentralized control approaches.

2.5.3 Cooperative Control of WMRs Based on MPC

Most recently, and due its ability to handle constraints and optimization, MPC has paid more attentions in the field of formation control of multiple robots. When applying MPC



(a) Centralized control (b) Decntralized control

Figure 2.11: Comparison between centralized and decentralized control approaches

to a team of robots, two approaches are considered, a centralized MPC and a decentralized/distributed MPC.

In the centralized MPC, a group of cooperative robots receives the control commands from one centralized decision maker; it may be either a robot in the formation or a ground station (as mentioned in Section 2.5.2). A single cost function established for the whole formation, the complete system is modeled, and all the control inputs are computed in one optimization problem for the entire robots system. With this strategy, the size of the state variables depends typically on the number of robots. The control horizon becomes larger, and the number of design variables, of which the system has to find their value, increases rapidly. Although short prediction horizons are desirable from a computational point of view, long prediction horizons are required for closed-loop stability and good performance. Therefore, the main disadvantage of using a centralized MPC is the huge computational effort due to the required on-line optimization. As a result, applying centralized MPC is so limited in real-time missions.

Decentralized/distributed MPC is proposed to avoid the computational effort problem of the centralized MPC (Dunbar and Murray, 2006; Chen et al., 2010; Wang et al., 2011). The idea is the same as the classical decentralized control (mentioned in Section 2.5.2) but with increasing the overall performance, based on applying MPC to each robot in the team. Each robot has the ability to make its own decision, taking into account its objective in addition to the objective of the whole team. Decentralized/distributed MPC has the advantage of combining the merits of decentralized control and MPC.

2.6 Fault Tolerant Cooperative Control (FTCC)

WMRs are designed to achieve their missions with higher efficiency, and with more safety and reliability. To achieve this objective, FTCC algorithms are designed to maintain safe operation and tolerate the effect of components malfunctions. Since this section is related to FTCC, then some basic definitions should be clarified.

Definition 2.6 (Fault). *A fault is an unpermitted deviation of at least one characteristic property (feature) of the system from the acceptable, usual, standard condition (Isermann,* *2006).*

Based on this definition, a fault corresponds to an abnormal behavior, in which it may not affect the overall function of the robot, but may eventually lead to failure. A fault may be small or hidden, and therefore undetectable. Faults can be classified into actuator, sensor, and system faults.

Actuator faults are usually characterized as the deviation between the expected actuator output and the actual one. They may result from aging or worn-out of actuator components, which leads to a bias or loss of effectiveness of the actuator. In addition, stuck of the actuator elements at a fixed value is one of the reasons of actuator faults. If the actuator stuck, any change in the actuator input will not cause any response at the actuator output. As a result, the control system of this actuator can be considered as an open-loop control system. Since the actuator is the component that carry out the control actions in a control system, faults in the actuator will have significant impact to the performance of the entire system.

Sensor faults refer to those situations that result in incorrect readings of the sensors. They may be due to aging, loss of effectiveness in the sensory elements, or unknown bias of the sensor output resulting from poor calibration or drift from their originally calibrated operating conditions, Sensor faults may also happened due to unexpected changes in the dynamic characteristics of the sensory elements, or broken connections in the sensory circuits. Since the signal from sensors often carry the most crucial information in any feedback control system, the state of the health of sensors is extremely important for a reliable operation of the controlled system.

From the control point of view, sensor is considered as a passive element in the control system. Therefore, using multiple sensors is one of the common strategies to increase the reliability of the sensory measurements. Unlike sensor, actuator is an active element in the control system, it carry out the control action to the system. Therefore, it is difficult to use multiple actuators for the same control signal. Consequently, this makes the design of FTCS more difficult.

Plant or system faults are caused by changes in the plant parameters or its dynamic characteristics due to aging or worn-out of its components. Therefore, plant/system faults are generally described as large changes in system parameters.

Definition 2.7 (Failure)**.** *A failure is a permanent interruption of a system's ability to perform a required function under specified operating conditions (Isermann, 2006).*

Based on the above definition, a failure is an event that terminates the functioning of a unit in the system. So, an actuator is declared failed when it cannot be controlled any more.

Most Recently, FTCC has received researchers' attention. In addition to maintaining a formation shape during task executions under normal conditions, it would be great that robots possess a fault-tolerance capability in the presence of faults. Thus, the healthy robots can react correspondingly to eliminate the negative effect on the mission completion. Otherwise, the formation will be broken while the mission accomplishment becomes impossible (Chamseddine et al., 2012).

FTCC can be achieved by appropriately re-assigning the task and re-coordinating the motion. Two cases need to be taken into account within the FTCC scheme:

- The robot cannot accomplish its assigned task in the presence of a severe fault. Thus, the mission should be re-assigned to the remaining healthy robots. As a result, the formation shape should be reconfigured according to the new assignment; and
- The robot can still complete the mission with degraded performance in the presence of faults. In this case, the other healthy robots will reconfigure their controllers considering the capability of the faulty robot.

During a formation mission, it is important for the robots to independently detect and isolate faults. Therefore, each robot should have its FDD unit which aims to detect the abnormal behaviors due to a component fault, determine the fault type and the exact location of the failed component, and identify the severity of the fault (Chamseddine et al., 2014). Then, the robot controller can be reconfigured to accommodate the fault based on the up-to-date information obtained from the FDD unit.

The existing FDD approaches can be classified into model-based and model-free (data-based) approaches. Since most of control techniques are model-based, then model-based approaches are the most common. The general idea of the model-based FDD approach is that when a component of a system fails, the system behavior will change from the nominal case. By measuring relevant signals on the system, it is possible to observe this behavior change and thereby infer that a fault has occurred. If the specific component that has failed should be identified, it is necessary to have a good description of the system such that the faulty behavior can be related to the component.

FDD and fault-tolerant control (FTC) techniques have been extensively investigated over the last three decades (see Zhang and Jiang, 2008; Yu and Jiang, 2015, and the references therein). Several interesting results have been presented for a WMR. In (Dixon et al., 2001), the kinematic and dynamic models of a WMR in the event of faults such as a change in wheel radius, or general kinematic disturbances induced by slipping and skidding faults is presented. A torque filtering technique is applied to develop a prediction error based fault detection signal. In (Duan et al., 2006), an adaptive particle filter is developed to detect and diagnose sensor faults for WMR, and experimental results are presented. Roumeliotis et al. (1998) developed a bank of Kalman filters for FDD purposes. However, FTC design is not addressed in this work. Further references related to FDD and FTC of WMRs under sensor and actuator faults can be found in (Duan et al., 2005).

FTCC has not yet been fully investigated in the literature. In (Mead et al., 2009), the authors present an FTCC strategy under actuator faults of terrestrial robots. In this study, the formation recovery according to fault occurrence is achieved through an auction-based algorithm. In (Chamseddine et al., 2012), the FTCC problem for a team of UAVs are considered. The formation recovery algorithm is proposed based on a trajectory re-planning technique. In the fault-free case, a differential flatness method is applied for each vehicle to plan its trajectory. Once an actuator fault occurs, by virtue of a centralized manner, a formation supervisor commands all the UAVs to re-plan their trajectories within the physical constraints of the faulty vehicle. In (Thumati et al., 2012), the FTCC of a team WMRs is investigated. Faults are counteracted by adding an extra term to the basic control law, which is a function of the fault dynamics, and recovered by a neural network. This work focuses on determining how to tolerate the individual fault of the team members; nevertheless the formation reconfiguration is omitted. In (Xu et al., 2014), an adaptive FTC algorithm for a team of UAVs subject to permanent and intermittent actuator faults is presented. The fault is modeled as a disturbance signal. The fault is estimated by an observer and therefore accommodated by a compensator to be added into the normal controller. However, the formation reconfiguration is not addressed. In (Yang et al., 2015), if an actuator fault occurs in a team of UAVs, then the other healthy members start reconfiguring the formation to reach the target point. In (Yu et al., 2016), FTCC scheme is considered for multiple UAVs. Feasible references in response to actuator faults can be generated by considering the health status of the team. While the FTCC gains can converge within finite time to facilitate the fault accommodation by applying the auxiliary integrated regressor matrix and vector method. This work can be seen as a motion re-coordination technique to keep the formation in presence of actuator fault.

From the existing literature, the following points are summarized for the FDD and FTCC problems of WMRs:

- Most of FDD algorithms developed for WMRs are mainly focus on sensor faults, not actuator faults.

- The existing FTCC studies mainly focus on the communication faults (Izadi et al., 2013; Yang et al., 2014), the obstacle avoidance problem to avoid the collision of the faulty robot (Lie and Go, 2011; Saber and Murray, 2003), and FTC of the individual team members (Thumati et al., 2012);

- Very few works investigate the case of severe actuator fault occurrence and determine how to accommodate the fault effect on the whole formation configuration (Yang et al., 2015). This case has two main challenges: how to generate the new formation configuration, and how to reconfigure the formation to the new one;

- Many studies focus on motion re-coordination (Chamseddine et al., 2012; Yu et al., 2016); and

- Most of the proposed approaches have not yet been validated in the real-time experiments.

During formation reconfiguration, sudden changes have occurred to the robots' control inputs. If these inputs are not bounded, the robots behavior become unstable, the formation may be broken, the and the robots may collide. Therefore, recently formation reconfiguration is formulated as an optimal control problem considering dynamic and algebraic constraints, hence intelligence optimization methods can be utilized to obtain the optimal solution (Duan et al., 2013). In (Furukawa et al., 2003) a time optimal control of multiple WMRs is presented based on the CPTD. Since time is assumed to be fixed in this work, it is not always possible to obtain the optimal solution within the given time. In (Xiong et al., 2007), an improved genetic algorithm (GA) for formation reconfiguration of multiple UAVs is developed, while time is an optimization parameter to be minimized. In (Duan et al., 2013), an approach combining PSO and GA is exploited for formation reconfiguration of multiple UAVs, within the control inputs, the collision avoidance and communications constraints. it is noted that all the optimal formation reconfiguration approaches in the above mentioned references have not yet validated in the real-time experiments.

2.7 Experimental Setup

The development of control algorithms for trajectory tracking and cooperative control of WMRs are increasing rapidly in the last decades. The simulations and real-time implementation validate the ability of the proposed control algorithms to achieve the desired performance, and guarantee the stability, safety and reliability.

In order to validate the proposed control algorithms presented in this thesis, these control strategies are implemented on a team of WMRs. This work is performed in the Networked Autonomous Vehicles Laboratory (NAV Lab) in the Department of Mechanical and Industrial Engineering, Concordia University. The experimental testbed includes:

- Quanser ground vehicles (QGV) with its control architecture;
- A ground station PC with QuaRC software used as a high level controller and wireless network; and
- The vision system.

Since the experiments are taking place indoor in the absence of GPS, so the high-level controller implemented on a ground station PC receives the states of the QGVs from the OptiTrack camera system. The high-level controller uses this information to calculate the desired pulse width modulation (PWM) to be sent to the QGVs' driving motors. Figure

2.12 shows the setup of the laboratory.

In all numerical simulations and experimental testing cases, the units used are m for position, deg for orientation, m/s for linear velocity, and rad/s for angular velocity. The desktop PC used in numerical simulations has a processor Intel(R) Core(TM) i7-3770 CPU @ 3.40 GHz, 4 GB RAM, and the operating system is Windows 7 Enterprise 64-bit. All the videos of the experiments performed in this thesis can be found in the NAV Lab YouTube Channel (NAV Laboratory, 2016).

2.7.1 QGV Description and its Control System Architecture

The differentially-driven WMR available in the NAV Lab is the Quanser QGV as shown in Figure 2.13. Quanser QGV is a two wheeled differentially-driven robot with a four degrees of freedom robotic manipulator. The front wheels of radius r = 7.8 cm are active and independent in performing driving and steering of the mobile robot. They are mounted on an axle of length h = 40 cm. The rear wheel is a dummy one and added at 30 cm from the front axle to improve robot's stability.

The QGV control module is comprised of a data acquisition board (HiQ DAQ) and an embedded Gumstix computer where QuaRC is the Quanser's real-time control software. Together with the Gumstix embedded computer, the HiQ controls the vehicle by reading

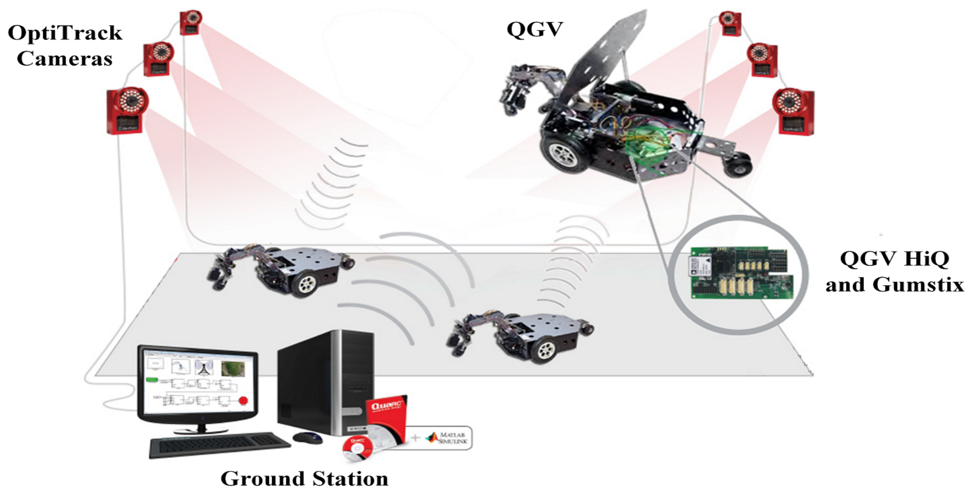


Figure 2.12: Laboratory setup

on-board sensors and sending motor commands. The motor speed controller is connected to two PWM servo outputs on the HiQ. The on-board Gumstix computer runs QuaRC, which allows rapidly developing and deploying controllers designed in MATLAB/Simulink environment for QGV real-time control. Runtime sensors measurement, data logging, and parameter tuning are supported between the ground host computer and the QGV (Quanser, Inc., 2012).

2.7.2 Ground Station

The ground station consists of a PC with the QuaRC and MATLAB/Simulink Softwares. The PC used to communicate with the robots and the sensors. Through a wireless network, the ground station communicates with the WMRs, where the PC works as a high-level controller, it generates the optimum control inputs and convert it to the desired PWM, then send it via the wireless link to the low-level controllers implemented on the robots (HiQ DAQ and the Gumstix on-board computer). The PC also connected to the OptiTrack camera system through a USB cables to collect the robots' motion states during the experiments. The desktop PC used in the experiments has a processor Intel(R) Core(TM) i5-5460 CPU @ 3.20 GHz, 8 GB RAM, and the operating system is Windows 7 Enterprise 64-bit.

2.7.3 Vision System

Twenty-four V100:R2 cameras which offer integrated image capture, processing, and motion tracking in a compact package constitute the OptiTrack's optical motion tracking system.

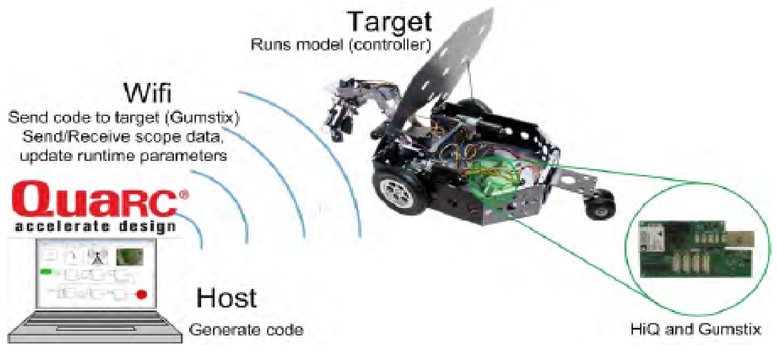


Figure 2.13: Quanser QGV with the communication module

The capability of customizing cameras with user-changeable M12 lenses and OptiTrack's exclusive Filter Switcher technology cameras has let V100:R2 cameras deliver one of the world's premier optical tracking value propositions. The V100:R2 is capable of capturing fast moving objects with its global shutter imager and 100 FPS capture speed. By maximizing its 640 $\times$ 480 VGA resolution through advanced image processing algorithms, the V100:R2 can also track markers down to sub-millimeter movements with repeatable accuracy (NaturalPoint, Inc., 2016). The OptiTrack system is connected to the ground station via USB cables. Figure 2.14 shows the V100:R2 camera.

Figure 2.14: V100:R2 camera used for measuring robots' states (NaturalPoint, Inc., 2016)

Chapter 3

Cooperative Control and Obstacle Avoidance of Multiple WMRs

This chapter explains the development of a control algorithm based on a combination of dynamic feedback linearization and LMPC. This algorithm is able to solve the trajectory tracking problem as well as the cooperative control of a team of WMRs. As mentioned in Section 2.5.1, applying an NMPC with WMRs has a problem of computational time in real-time implementation. Nevertheless, the applied LMPC in the literature was time-varying, results in almost solving the computational time problem in a single WMR case, but with increasing the number of robots, the computational burden problem still exists.

Using feedback linearization, the nonlinearity problem is solved and robot model becomes linear time invariant (LTI) with new control inputs. On the other hand, LMPC is applied to the linearized model to perform the trajectory tracking as well as the entire formation mission. The main advantage is that the robot model becomes LTI, so applying MPC with multiple robots becomes possible and the computational time problem can be overcame.

This chapter is organized as follows. First, feedback linearization of WMRs and developing the linearized model of the robot is presented. Next, the proposed control algorithm is applied to achieve the trajectory tracking of a single WMR, and a stability analysis of the control algorithm is illustrated. Following, formation control of multiple WMRs based on the proposed algorithm is applied in a distributed manner. Then, an obstacle avoidance algorithm is added to the proposed control algorithm. In addition, a cooperative control of

UAVs and WMRs for forest monitoring and fire detection is presented to emphasize the effectiveness of the proposed formation control algorithm in such a real life application. Finally, to validate the effectiveness of the proposed algorithms, simulation and experimental results are presented.

3.1 Feedback Linearization of WMRs

Feedback linearization is a common approach used with nonlinear systems. The concept is to make use of algebraic transformation of nonlinear system dynamics to an equivalent linear system, so that linear control laws can be applied to nonlinear systems. Generally speaking, there are two types of linearization (Tzafestas, 2014):

- *Input-state linearization*: In this case, the required is to find a state transformation $z = z(x)$ and an input/transformation $u = u(x, v)$, where v is the new manipulable input. The purpose of this transformation is to bring the system $\dot{x} = f(x, u)$ to the linear system $\dot{z} = Az + Bv$; and

- *Input-output linearization*: in which the basic idea is if one has a system $\dot{x} = f(x, u)$ with output $y = h(x)$, then the basic feature of this system is that the output y is connected to u only indirectly through x. Therefore, to achieve input-output linearization, one must find a direct relationship between the input and the output of the system. This may be done by successive differentiation of the output until all inputs appear in the resulting derivative equations.

Feedback linearization was used for solving the trajectory tracking problem of WMRs. In (Oriolo et al., 2002), feedback linearization is applied to the kinematic model of the robot, while a proportional derivative (PD) control law is used to solve the trajectory tracking problem. In (Shojaei et al., 2009), a combination of adaptive control and feedback linearization are applied for a single robot trajectory tracking. In (d'Andréa Novel et al., 1995), a trajectory tracking problem of different types of WMRs solved by means of feedback linearization based on static and dynamic state feedback laws. In (Chwa, 2010), the author proposed a tracking controller using a backstepping-like feedback linearization. They applied the feedback linearization in for both the kinematic and the dynamic models in a cascaded manner.

3.1.1 Input-State Linearization of WMRs

As mentioned in Section 2.3.2, differentially-driven WMR kinematic model presented in (2.4) belongs to the driftless affine systems that described by the following form:

$$\dot{q} = \sum_{i=1}^{m} g_i(q) u_i \tag{3.1}$$

Theorem 3.1. *System (3.1) is not input-state linearizable by a smooth state feedback.*

Proof. If the system is input state linearizable, then it has to satisfy the following two conditions: the accessibility rank condition and the involutivity condition (Nijmeijer and der Schaft, 1990). As mentioned in Section 2.3.2, the system (3.1) satisfied the accessibility rank condition. Now, we will show that this system does not satisfy the involutivity condition.

As mentioned in Section 2.3.2, differentially-driven WMRs have two vector fields g_1 and g_2 where:

$$g_1 = \begin{bmatrix} \cos\phi \\ \sin\phi \\ 0 \end{bmatrix}, \quad g_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

and it results the following two-dimensional distribution:

$$\Delta = \{g_1, g_2\} = \left\{ \begin{bmatrix} \cos\phi \\ \sin\phi \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}$$

This distribution is nonsingular because for any (x, y, ϕ), the resulting vector space $\Delta(x, y, \phi)$ is two-dimensional. The involutivity of Δ can be found by finding the Lie bracket of g_1 and g_2 as mentioned in (2.13), and the result is the matrix

$$\overline{\Delta} = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ \sin\phi & 0 & -\cos\phi \\ 0 & 1 & 0 \end{bmatrix}$$

which has $\det \overline{\Delta} = 1 \neq 0$ and rank $= 3 \geq 2$. Then, the third vector field (results from the Lie bracket) is linearly independent of g_1 and g_2. Consequently the distribution Δ is not involutive. $\square$

3.1.2 Input-Output Linearization of WMRs

Although the differentially-driven WMR is not input-state linearizable, it may be input-output linearizable and decoupled depending on the choice of the outputs. Two types of feedback are commonly employed for the purpose of linearization: static state feedback and dynamic state feedback. The dynamic state feedback is more general and includes the static state feedback as a special case. Consequently, the conditions for the dynamic state feedback are more complicated (Yun and Yamamoto, 1992).

For a differentially-driven WMRs, they are input-output linearizable with any static feedback of the form:

$$\varphi = \alpha(q) + \beta(q)u \tag{3.2}$$

However, for exact linearization, the input-output linearization may be achieved by using a dynamic feedback linearization (Yun and Yamamoto, 1992; Oriolo et al., 2002).

For a driftless nonlinear system (2.4) which can be re-written in the following compact form:

$$\dot{q} = S(q)\varphi \qquad q \in \mathbb{R}^{\varepsilon}, \varphi \in \mathbb{R}^{m} \tag{3.3}$$

the input-output feedback linearization can be achieved by using a dynamic feedback compensator of the form (Oriolo et al., 2002):

$$\begin{aligned} \dot{\xi} &= a(q,\xi) + b(q,\xi)u \\ \varphi &= c(q,\xi) + d(q,\xi)u \end{aligned} \tag{3.4}$$

with state $\xi \in \mathbb{R}^n$ and input $u \in \mathbb{R}^m$. Therefore the system (3.3) and (3.4) is equivalent to a linear system under a state transformation $z = T(q, \xi)$.

The first step is the definition of an output vector δ, with $\delta \in \mathbb{R}^m$. Since a differentially-driven WMR has two inputs v and ω, it is natural to select an output equation with its two independent components, the coordinates of point q as $\delta = [x, y]^T$. Following the steps presented in (Oriolo et al., 2002), and start by differentiating with respect to time yields:

$$\dot{\delta} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos\phi & 0 \\ \sin\phi & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.5}$$

Equation (3.5) shows that only v affects $\dot{\delta}$, while the angular velocity ω cannot be recovered from this first-order differential information. To proceed, adding an integrator ξ on the linear velocity input, thus

$$v = \xi, \qquad \dot{\xi} = a, \tag{3.6}$$

where a is the linear acceleration of the WMR. Differentiating again, one obtains

$$\ddot{\delta} = \dot{\xi} \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} + \xi\dot{\phi} \begin{bmatrix} -\sin\phi \\ \cos\phi \end{bmatrix} = \begin{bmatrix} \cos\phi & -\xi\sin\phi \\ \sin\phi & \xi\cos\phi \end{bmatrix} \begin{bmatrix} a \\ \omega \end{bmatrix} \tag{3.7}$$

Assuming that the matrix multiplying the modified input (a,ω) is nonsingular if $\xi \neq 0$, then it is obtained that:

$$\begin{bmatrix} a \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\phi & -\xi\sin\phi \\ \sin\phi & \xi\cos\phi \end{bmatrix}^{-1} \ddot{\delta} \tag{3.8}$$

Defining $\ddot{\delta}$ as

$$\ddot{\delta} = \begin{bmatrix} \ddot{\delta}_1 \\ \ddot{\delta}_2 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = u \tag{3.9}$$

where u is the new control input vector. Then, equation (3.8) can be written as:

$$\begin{bmatrix} a \\ \omega \end{bmatrix} = \frac{1}{\xi} \begin{bmatrix} \xi\cos\phi & \xi\sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{3.10}$$

The resulting dynamic compensator is:

$$\begin{aligned} \dot{\xi} &= u_1\cos\phi + u_2\sin\phi \\ v &= \xi \\ \omega &= \frac{u_2\cos\phi - u_1\sin\phi}{\xi} \end{aligned} \tag{3.11}$$

Then, the new coordinates are:

$$\begin{aligned} z_1 &= x \\ z_2 &= y \\ z_3 &= \dot{x} = \xi \cos\phi \\ z_4 &= \dot{y} = \xi \sin\phi \end{aligned} \tag{3.12}$$

The above equation can be rewritten as:

$$\begin{aligned} \ddot{z}_1 &= u_1 \\ \ddot{z}_2 &= u_2 \end{aligned} \tag{3.13}$$

Representing the new linearized model as a state space system as $\dot{z} = A_d z + B_d u$, then the equivalent linear model is:

$$\dot{z} = \begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \\ \dot{z}_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{3.14}$$

where $z \in \mathbb{R}^{n_l}$ and $u \in \mathbb{R}^{m_l}$ are the vectors of the robot's states and control inputs, respectively. $A_d \in \mathbb{R}^{n_l} \times \mathbb{R}^{n_l}$, and $B_d \in \mathbb{R}^{n_l} \times \mathbb{R}^{m_l}$ are the state and input matrices, respectively. n_l and m_l are the number of states and control inputs, respectively.

The dynamic compensator (3.11) has a singularity at $v = 0$. This singularity in the dynamic extension process is structural for nonholonomic robots (De Luca and Benedetto, 1993). This singularity can be avoided when designing the control laws on the linear model. Next, a sufficient conditions for avoiding the singularity are presented based on Theorem 1 in (Oriolo et al., 2002).

Theorem 3.2. *Let a nonholonmic robot* (2.4) *and* (3.3) *is input-output linearizable by means of a dynamic feedback linearization. Then a dynamic compensator* (3.11) *is applied. Let the nonholonomic robot track a reference trajectory $x_r(t), y_r(t)$, then the tracking error components are:*

$$x_e = x_r - x, \qquad y_e = y_r - y$$

Let, λ_{i1},λ_{i2} *with* $i = \{1, 2\}$*, be the eigenvalues of the closed-loop dynamics of the error components. Assume that* $\lambda_{i1} < \lambda_{i2} < 0$ *and* λ_{i2} *is sufficiently small. Then if*

$$\min_{t\geq 0} \left\| \begin{pmatrix} \dot{x}_r(t) \\ \dot{y}_r(t) \end{pmatrix} \right\| > \left\| \begin{pmatrix} \dot{x}_e(0) \\ \dot{y}_e(0) \end{pmatrix} \right\| \tag{3.15}$$

with $\dot{x}_e(0) \neq 0$ *and* $\dot{y}_e(0) \neq 0$*, then the singularity* ξ *will be avoided*

Proof. For

$$|\xi| = \left\| \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \right\| = \left\| \begin{pmatrix} \dot{x}_r - \dot{x}_e \\ \dot{y}_r - \dot{y}_e \end{pmatrix} \right\| \geq \left\| \begin{pmatrix} \dot{x}_d \\ \dot{y}_d \end{pmatrix} \right\| - \left\| \begin{pmatrix} \dot{x}_e \\ \dot{y}_e \end{pmatrix} \right\| \tag{3.16}$$

$\xi = 0$ is avoided if

$$\min_{t\geq 0} \left\| \begin{pmatrix} \dot{x}_r(t) \\ \dot{y}_r(t) \end{pmatrix} \right\| > \left\| \begin{pmatrix} \dot{x}_e(\tau) \\ \dot{y}_e(\tau) \end{pmatrix} \right\|, \qquad \forall \tau \geq 0 \tag{3.17}$$

Using the solution of the closed loop error dynamics

$$\dot{x}_e(t) = a_{11}e^{\lambda_{11}t} + a_{12}e^{\lambda_{12}t}$$
$$\dot{y}_e(t) = a_{21}e^{\lambda_{21}t} + a_{22}e^{\lambda_{22}t}$$

where a_{jk} with $j = \{1, 2\}$ and $k = \{1, 2\}$ are constants depend on the chosen eigenvalues, it is shown that the norm of the velocity error is upper bounded by its value at $t = 0$. □

3.2 Single Robot Trajectory Tracking

A combination of the feedforward action with a feedback control is required to solve the trajectory tracking problem. This section presents how to solve the trajectory tracking problem for a single robot based on the combination of both feedforward and feedback actions.

3.2.1 Feedforward Commands Generation

Assume a given reference trajectory $(x_r(t), y_r(t))$ defined in a time interval $t \in [0, T]$, then the feedforward command generation involves generating the desired reference velocities $v_r(t)$

and $\omega_r(t)$ as follows:

$$v_r(t) = \pm\sqrt{\dot{x}_r^2(t) + \dot{y}_r^2(t)} \tag{3.18}$$

$$\omega_r(t) = \frac{\ddot{y}_r(t)\dot{x}_r(t) - \ddot{x}_r(t)\dot{y}_r(t)}{\dot{x}_r^2(t) + \dot{y}_r^2(t)} \tag{3.19}$$

where the sign $\pm$ depends on the robot direction,i.e. $+$ for the forward motion, - for the reverse motion.

The reference angle $\theta_r(t)$ can be defined as:

$$\phi_r(t) = \arctan 2(\dot{y}_r(t), \dot{x}_r(t)) + K_d\pi, \quad K_d = 0, 1 \tag{3.20}$$

where $\arctan 2$ is the four-quadrant inverse tangent function. K_d defines the driving direction. 0 for forward motion, and 1 for reverse.

3.2.2 Feedback Control Design

After obtaining the linearized model based on feedback linearization, the trajectory tracking controller is applied using an LMPC strategy to simply the controller design for such WMR nonlinear system. Figure 3.1 presents the overall control system block diagram. The outputs of the LMPC are the optimum values of the control inputs u_1 and u_2. These signals should be fed to the dynamic compensator (3.11) in order to obtain the actual control inputs v and ω which fed to the robot to track the reference trajectory.

Writing the system (3.14) in discrete form as:

$$z(k+1) = Az(k) + Bu(k) \tag{3.21}$$

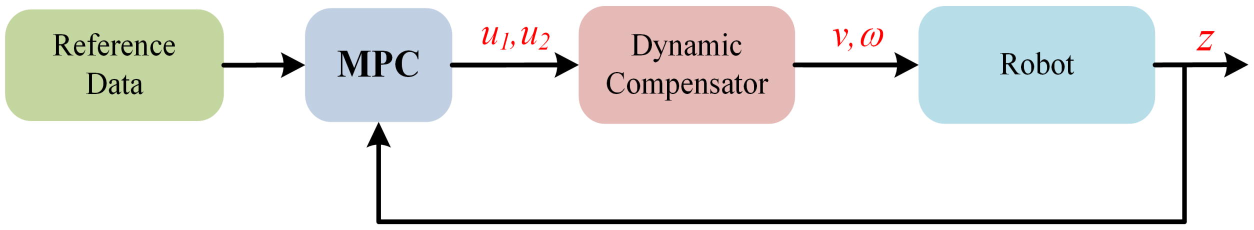


Figure 3.1: Control system block diagram for a single robot

with:

$$
\begin{aligned}
A &= I + A_d T_s, \\
B &= B_d T_s
\end{aligned}
$$

where T_s is the sampling time, $A \in \mathbb{R}^{n_l} \times \mathbb{R}^{n_l}$, and $B \in \mathbb{R}^{n_l} \times \mathbb{R}^{m_l}$ are the state and input matrices, respectively.

Assuming that all states are measurable, then the objective function for a robot J to be minimized can be stated as a quadratic cost function of the states and the inputs as:

$$
\min_{u(.)} J = \sum_{i=1}^{N_p-1} z_e^T(k+i|k) Q z_e(k+i|k) + \sum_{i=0}^{N_c} u^T(k+i|k) R u(k+i|k) \tag{3.22}
$$

subject to

$$
\begin{aligned}
& z_e(k+i|k) = A z_e(k+i-1|k) + B u(k+i-1|k), \\
& z_e(k+i|k) \in \mathcal{Z}, \\
& u(k+i|k) \in \mathcal{U}
\end{aligned} \tag{3.23}
$$

where z_e is state error to be minimized, N_p and N_c denote both the prediction and control horizons respectively. $Q \in \mathbb{R}^{n_l} \times \mathbb{R}^{n_l}$ and $R \in \mathbb{R}^{m_l} \times \mathbb{R}^{m_l}$ are the weighting matrices with $Q > 0$ and $R > 0$. $\mathcal{Z} \subset \mathbb{R}^{m_l}$ is the state constraints. $\mathcal{U} \subset \mathbb{R}^{n_l}$ is the input constraints. Usually, $\mathcal{U} = \{u \in \mathbb{R}^{n_l} : u_{min} \leq u \leq u_{max}\}$. u_{min} and u_{max} are known constants in $\mathbb{R}^{m_l}$. The first term on the right hand side of equation (3.22) is called the state penalty and the second term is called control penalty.

The control goal is to find $u(k)$ which drives the system (3.21) toward the equilibrium in which $z_e(k) = 0$ and $u(k) = 0$.

To re-write the objective function in a quadratic form, the following vectors can be defined:

$$
\hat{z}_e(k+1) = \begin{bmatrix} z_e(k+1|k) \\ z_e(k+2|k) \\ \vdots \\ z_e(k+N_p|k) \end{bmatrix}, \quad \hat{u}(k) = \begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u(k+N_p-1|k) \end{bmatrix}
$$

where $\hat{z}_e \in \mathbb{R}^{n_l \times N_p}$ and $\hat{u} \in \mathbb{R}^{m_l \times N_p}$. Equation (3.22) can be written as:

$$J(k) = \hat{z}_e^T(k+1)\hat{Q}\hat{z}_e + \hat{u}^T(k)\hat{R}\hat{u} \tag{3.24}$$

with

$$\hat{Q} = \begin{bmatrix} Q & 0 & \dots & 0 \\ 0 & Q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Q \end{bmatrix}, \quad \hat{R} = \begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & R \end{bmatrix}$$

where $\hat{Q} \in \mathbb{R}^{n_l \times N_p} \times \mathbb{R}^{n_l \times N_p}$ and $\hat{R} \in \mathbb{R}^{m_l \times N_p} \times \mathbb{R}^{m_l \times N_p}$.

So, from (3.21), $\hat{z}_e(k+1)$ can be written as:

$$\hat{z}_e(k+1) = \hat{A}(k)z_e(k|k) + \hat{B}(k)\hat{u}(k) \tag{3.25}$$

with

$$\hat{A}(k) = \begin{bmatrix} A(k|k) \\ A(k|k)A(k+1|k) \\ \vdots \\ \rho(k,0) \end{bmatrix}$$

and

$$\hat{B}(k) = \begin{bmatrix} B(k|k) & 0 & \dots & 0 \\ A(k+1|k)B(k|k) & B(k+1|k) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \rho(k,1)B(k|k) & \rho(k,1)B(k+1|k) & \dots & B(k+N_p-1|k) \end{bmatrix}$$

where $\hat{A} \in \mathbb{R}^{n_l \times N_p} \times \mathbb{R}^{n_l}$ and $\hat{B} \in \mathbb{R}^{n_l \times N_p} \times \mathbb{R}^{m_l \times N_p}$, and $\rho(k,r)$ is defined as:

$$\rho(k,r) = \prod_{r=i}^{N_p-1} A(k+r|k)$$

Finally, the objective function can be written in a standard quadratic form as:

$$J(k) = 0.5\hat{u}^T(k)H(k)\hat{u}(k) + f^T(k)\hat{u}(k) \tag{3.26}$$

where

$$H(k) = 2(\hat{B}^T(k)\hat{Q}\hat{B}(k) + \hat{R}), \quad H(k) \in \mathbb{R}^{m_l \times N_p} \times \mathbb{R}^{m_l \times N_p}$$

$$f(k) = 2\hat{B}^T(k)\hat{Q}\hat{A}(k)z(k|k), \quad f(k) \in \mathbb{R}^{m_l \times N_p}$$

The matrix H is the Hessian matrix, and it must be positive definite. It presents the quadratic part of the objective function, while the vector f represents the linear part.

Remark 3.1. *In real-time implementation, $\dot{x}$ and $\dot{y}$ are calculated by numerically differentiating (x, y).*

Remark 3.2. *The necessary condition for combining both the feedforward and the feedback control actions is that $v_r(t) \neq 0$. If it happened at a specific time t, the robot will rotate with the angular velocity $\omega_r(t)$ (Klančar and Škrjanc, 2007).*

Remark 3.3. *For exact trajectory tracking, the initial value of the state of the dynamic compensator ξ should be equals to the initial value of the velocity of the reference trajectory v_r (Oriolo et al., 2002), i.e.*

$$\xi(0) = v_r(0) = \dot{x}_r(0)\cos\phi_r(0) + \dot{y}_r(0)\sin\phi_r(0)$$

Remark 3.4. *Using this method, it is not required to compute the desired orientation $\phi_r(t)$*

3.2.3 Stability Analysis

In this section, a stability analysis for the proposed MPC algorithm will be presented. The following theorem discuss the MPC stability.

Theorem 3.3. *Suppose the following assumptions hold for a nominal controller $L(z_e)$, a terminal state weight $\Omega(z_e)$ and a terminal state domain $\mathcal{Z}$*

1. $Az_e + BL(z_e) \in \mathcal{Z}, \quad \forall z \in \mathcal{Z}$

2. $0 \in \mathcal{Z}$

3. $\Omega(Az_e + BL(z_e)) - \Omega(z_e) \leq -z_e^T Q z_e - L^T(z_e)RL(z_e), \quad \forall z_e \in \mathcal{Z}$

4. $\Omega(0) = 0, \quad \Omega \succ 0$

5. $L(z_e) \in \mathcal{U}, \quad \forall z_e \in \mathcal{Z}$

Then, assuming the feasibility at the initial state, then the optimization problem presented in (3.22) will be updated to guarantee asymptotic stability as follow:

$$\min_{u(.)} J(k) = \sum_{i=1}^{N_p-1} z_e^T(k+i|k)Qz_e(k+i|k) + \sum_{i=0}^{N_c} u^T(k+i|k)Ru(k+i|k) + \Omega(z_e(k+N_p|k)) \quad (3.27)$$

subject to

$$\begin{aligned} &z_e(k+i|k) = Az_e(k+i-1|k) + Bu(k+i-1|k), \\ &z_e(k+i|k) \in \mathcal{Z}, \\ &u(k+i|k) \in \mathcal{U} \end{aligned} \quad (3.28)$$

Proof. Based on using the objective function $J(k)$ as a Lyapunov function, let us denote the optimal cost as $J^*(k)$. Then, the optimal cost at time k is obtained with the control sequence $[u^*(k|k)\ u^*(k+1|k)\ \ldots\ u^*(k+N_p-1|k)]$, where the notation * related to the optimal solution. So, at time $k+1$, the feasible solution is $[u^*(k+1|k)\ u^*(k+2|k)\ \ldots\ u^*(k+N_p-1|k)\ L(z_e^*(k+N_p|k))]$ where $z_e^*(k+N_p|k) \in \mathcal{Z}$. According to Assumption 5, it is clear that $L(z_e^*(k+N_p|k))$ satisfies the control constraint. Also Assumption 1 satisfy the terminal state constraint. Then the cost using this control sequence will be

$$\begin{aligned} J(k+1) = &\sum_{i=1}^{N_p-1} z_e^{*T}(k+i+1|k)Qz_e(k+i+1|k) + \sum_{i=0}^{N_c} u^{*T}(k+i+1|k)Ru(k+i+1|k) \\ &+ z_e^T(k+N_p|k)Qz_e(k+N_p|k) + L^T(z_e(k+N_p|k))RL(z_e(k+N_p|k)) \\ &+ \Omega(z_e(k+N_p+1|k)) \\ = &\sum_{i=1}^{N_p-1} z_e^{*T}(k+i+1|k)Qz_e(k+i+1|k) + \sum_{i=0}^{N_c} u^{*T}(k+i+1|k)Ru(k+i+1|k) \\ &+ \Omega(z_e^*(k+N_p|k)) + \Omega(z_e(k+N_p+1|k)) - \Omega(z_e^*(k+N_p|k)) \\ &+ z_e^{*T}(k+N_p|k)Qz_e(k+N_p|k) + L^T(z_e(k+N_p|k))RL(z_e(k+N_p|k)) \\ &- z_e^T(k|k)Qz_e(k|k) - u^{*T}(k|k)Ru^T(k|k) \end{aligned} \quad (3.29)$$

Some terms are added and subtracted to get an expression of $J(k+1)$ containing $J(k)$. The first line of last equality now is the optimal cost at time k, i.e., $J^*(k)$. According to

Assumption 3, the sum of the second and the third lines in the last equality is negative, so:

$$J(k+1) \leq J^*(k) - z_e^T(k|k)Qz_e - u^{*T}(k|k)Ru^*(k|k) \tag{3.30}$$

Since $J(k+1) \geq J^*(k+1)$, then

$$J^*(k+1) \leq J^*(k) - z_e^T(k|k)Qz_e - u^{*T}(k|k)Ru^*(k|k) \tag{3.31}$$

which means that hence x(k) converges to the origin, then $J^*(k)$ will be in a decaying sequence. □

More details of this proof can be found in (Mayne et al., 2000; Lofberg, 2000), also the methods to choose $\mathcal{Z}$, $L(z_e)$, and $\Omega(z_e)$ can be found in (Lofberg, 2000).

3.3 Formation Controller

After the design of the proposed control algorithm, and being implemented to a single robot. This algorithm is applied to a team of WMRs to perform a specific formation configuration.

Consider j^{th} robot in a team of robots moving in a specific formation within a leader-follower scheme, $j \in \{l, 1, 2, \ldots, N\}$ denotes the formation configuration of the leader l and N followers. The leader l should track a predefined trajectory $(x_r(t), y_r(t))$ defined in a time interval $t \in [0, T]$ (as mentioned in Section 3.2), while followers N should follow the leader maintaining a desired formation configuration $(l_d - \phi_d)$ between them and the leader as shown in Figure 3.2, where l_d and ϕ_d are the follower's relative distance and angle with respect to the leader.

Given the leader position, and as long as the formation configuration $l_d - \phi_d$ is given and fixed, then the followers' position will be unique. So, the formation among team members can be represented by $l_d - \phi_d$. To achieve the desired formation based on the algorithm presented in Section 3.2, it is necessary to find the followers' reference trajectories that they must follow to maintain the desired formation configuration between them and the leader based on the leader position and the desired formation data as shown in Figure 3.3.

As noticed, the $l_d - \phi_d$ configuration is a polar coordinates representation of the fol-

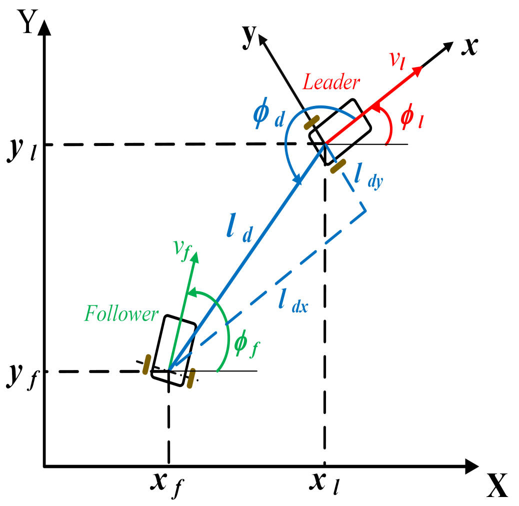


Figure 3.2: Leader-follower formation geometry

lower's relative position. So, to apply the proposed control algorithm, the follower's relative position should be represented in Cartesian coordinates (X,Y). Also, representing the formation control in polar coordinates will lead to a singularity (Li et al., 2004).

To represent the system in Cartesian coordinates, let l_{d_x} and l_{d_y} be the projection of l along x and y coordinates, thus:

$$l_{d_x} = -(x_l - x_f)\cos\phi_l - (y_l - y_f)\sin\phi_l, \tag{3.32}$$

$$l_{d_y} = (x_l - x_f)\sin\phi_l - (y_l - y_f)\cos\phi_l, \tag{3.33}$$

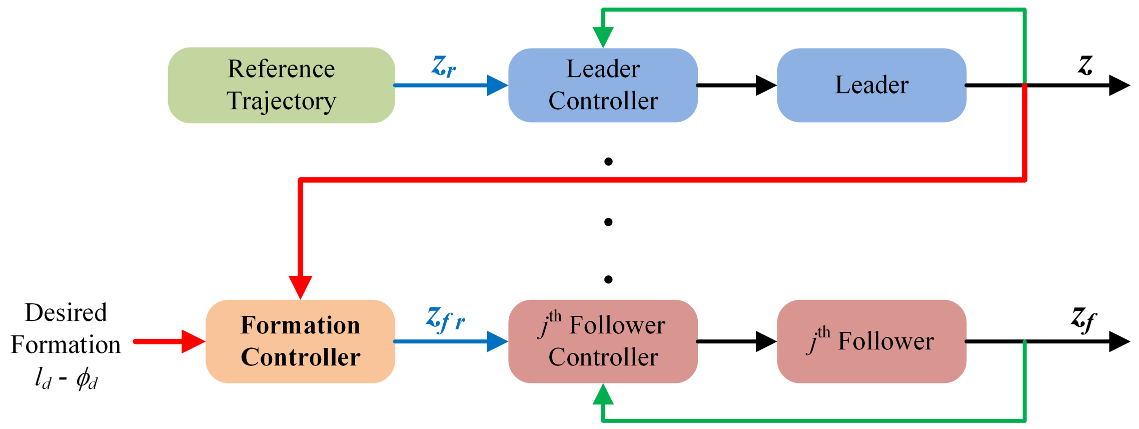


Figure 3.3: Formation control principle

also;

$$l_{d_x} = l_d \cos \phi_d, \tag{3.34}$$

$$l_{d_y} = l_d \sin \phi_d. \tag{3.35}$$

From Eqs. (3.32) and Eq. (3.33), the desired follower position will be:

$$x_{f_r} = x_l + l_{d_x} \cos \phi_l - l_{d_y} \sin \phi_l, \tag{3.36}$$

$$y_{f_r} = y_l + l_{d_x} \sin \phi_l + l_{d_y} \cos \phi_l. \tag{3.37}$$

and consequently:

$$\dot{x}_{f_r} = \dot{x}_l - l_{d_x} \sin(\phi_l)\dot{\phi}_l - l_{d_y} \cos(\phi_l)\dot{\phi}_l, \tag{3.38}$$

$$\dot{y}_{f_r} = \dot{y}_l + l_{d_x} \cos(\phi_l)\dot{\phi}_l - l_{d_y} \sin(\phi_l)\dot{\phi}_l. \tag{3.39}$$

To achieve the desired formation, each follower controller should be constrained by the following condition:

$$\lim_{t\to\infty}(z_{f_r}(t) - z_f(t)) = 0 \quad \forall\, N < j \tag{3.40}$$

where $z_{f_r}(t) = [x_{f_r}, y_f, \dot{x}_{f_r}, \dot{y}_{f_r}]^T$ that can be obtained from equations (3.36) to (3.39).

Now, the reference trajectory with each follower relative to the leader can be calculated, and then the proposed algorithm presented in Section 3.2 for a single robot can be applied to each robot in the team. Each robot uses its local information provided by its sensors as well as the leader position to accomplish the formation mission. Therefore, the leader-follower configuration can be implemented in a distributed manner based on the robots' local information.

3.4 Obstacle Avoidance Algorithm

Another challenge for formation control is to ensure that the robots should move safely in a semi-structured environment. During task execution, robots are expected to operate in a cluttered environment. Therefore, the robot must be equipped by a sensing module to indicate the location of nearby obstacles, and send these data to an obstacle avoidance algorithm.

This algorithm will get the robot to avoid the surrounding obstacles. Obstacle avoidance algorithms can be classified into motion planning control and reactive control (Rodríguez-Seda et al., 2014). In a motion planning control, the controller determines the obstacle-free trajectory that the robot must follow based on obstacles' positions. Therefore, this method is not efficient when the robot facing a moving obstacles. On the other side, reactive control strategy is based on the sensory information, so the control inputs are computed on-line as the obstacles are detected to avoid them. Moreover, using a reactive control strategies allow the robot to avoid the moving obstacles also. Therefore, reactive methods are preferred when working in dynamic environments. Several methods such as a potential field (Khatib, 1986), the tangential escape (Ferreira et al., 2006) and the mechanical impedance with virtual force (Rampinelli et al., 2010) are examples of the reactive control method that applied with WMRs. All these methods assumes a perfect obstacle sensing.

This section presents how the system can avoid the obstacles based on the sensory information about the surrounding environment. The following assumptions are asserted for the obstacle avoidance algorithm:

Assumption 3.1. *Each robot has an on-board sensing system that can detect the other robots based on distance measurements.*

Assumption 3.2. *The maximum range of sensors is greater than the distance d_{max} (see Figure 3.4).*

Assumption 3.3. *Robots' sensors have reliable sensing.*

The reactive obstacle avoidance algorithm presented here is similar to (Rampinelli et al., 2010), where the mechanical impedance principle is used. The main concept is to link the movement of each robot in formation to a virtual repulsive forces based on its interaction with the surrounding environment. Then, the linear and angular velocities of the robot are changed in response to this repulsive force. The value of that repulsive force can be given as follows:

$$F = \begin{cases} C(d_{max} - d)^H & \text{if } d < d_{max} \\ 0 & \text{if } d \geq d_{max}, \end{cases} \tag{3.41}$$

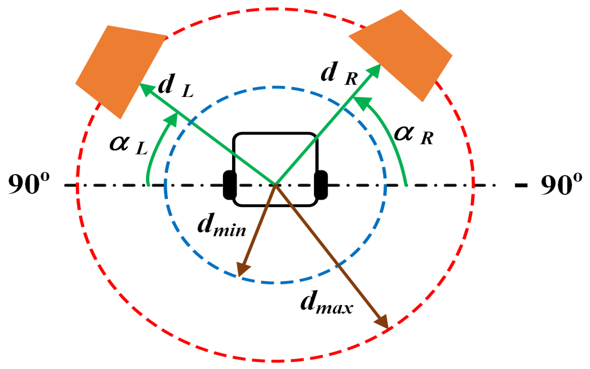


Figure 3.4: Robot-obstacles interaction during robot motion (Rampinelli et al., 2010)

where d is the actual distance between the robot and the closest obstacle, d_{max} is the maximum robot-obstacle distance, H is a positive integer, and C is a constant corresponding to system calibration and given by:

$$F_{max} = C(d_{max} - d_{min})^s, \tag{3.42}$$

where d_{min} is the minimum acceptable robot-obstacle distance to avoid the collision, s is a positive integer, and F_{max} is the maximum value of virtual force.

In case of robot movement in environment containing many obstacles, each robot generates two fictitious forces F_R and F_L that relate to the closest right and left obstacles to the robot, respectively. As observed in Figure 3.4, d_R, α_R, d_L and α_L are the distances and angles associated to the right and left obstacles, respectively. Replacing F and d in (3.41) by F_R and d_R, F_L and d_L, then the values both the right and left repulsive forces are calculated.

The relationship between the fictitious forces and the corrections in the linear and angular velocities v_c and ω_c respectively necessary to avoid the obstacles are given by:

$$\begin{aligned} v_c &= Z^{-1}(F_R \sin\alpha_R + F_L \sin\alpha_L), \\ \omega_c &= Z^{-1}(F_R - F_L), \end{aligned} \tag{3.43}$$

where Z represents the mechanical impedance of the environment defined as:

$$Z = Is^2 + Bs + K,$$

where I, B and K are positive constants representing the inertia, the damping and the elasticity of the dynamics of the robot-environment interaction, respectively. The whole control system block diagram for the team of robots including the obstacle avoidance algorithm is shown in Figure 3.5.

Remark 3.5. *The above mentioned strategy is applied to each robot in the formation individually as shown in Figure 3.5.*

3.5 Cooperative Control of UAVs-WMRs for Forest Monitoring and Fire Detection

In recent years, research in cooperative unmanned systems have received growing attention in both civilian and military applications. One of these applications is the forest monitoring and fire detection.

Forest fire detection becomes very important in the whole world, especially in North America (Yuan et al., 2015). Approximately 27 million acres have been consumed because of wild-land fires during 2005-2007 (Kumar et al., 2011). As a result, a lot of people have been displaced causing great financial loss. Apart from this socio-economic loss, smoke-related effects on human and wildlife are dominant concerns.

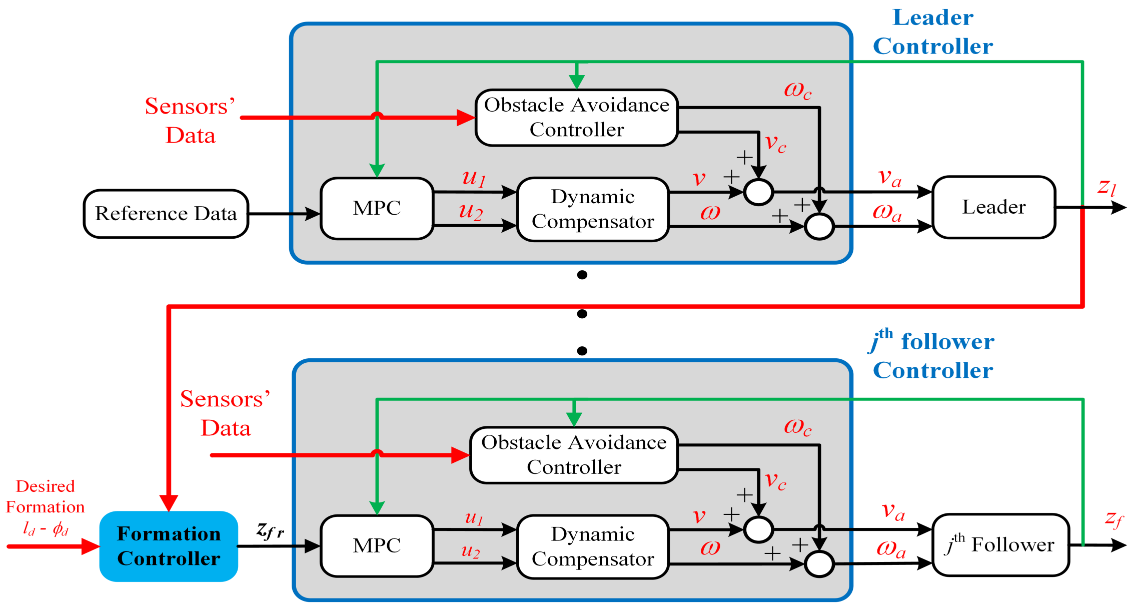


Figure 3.5: Overall control system block diagram for the whole team

UAVs can play a very important role in forest fire detection and fighting. They provide reliability in extreme operational conditions and monotonous repeated tasks for longer duration, enabling long-term data gathering and situational awareness (Kumar et al., 2011). Usually, UAVs have limited payload since the main features of UAV design is to be smaller with better maneuverability for easy use and operation. As a result, they have a limited battery life, and consequently limited running time. This problem arises especially with the rotary-wing UAVs due to its higher power consumption during mission execution. On the other hand, WMRs offer high payload and longer running time. Accordingly, WMRs can be used to transport UAVs from a central ground station to a safety spot close enough for UAVs to take-off for mission execution, and return to its landing platform on WMRs. During monitoring and fire detection mission, it is of great importance for UAVs to capture images using on-board CCD cameras, thermal and infrared cameras. These images need a powerful processor for on-line data analysis and image processing, which is difficult to embed on the UAVs due to their payload limitation. Therefore, WMRs can be used as local ground stations for UAVs during mission execution, since WMRs have more payload capabilities to carry the required equipment for the purpose of fire fighting.

To emphasize the effectiveness of the proposed formation control algorithm, it is implemented on a team of WMRs to pair with a team on UAVs for forest monitoring and fire detection application in order to avoid the problems of limited payload and running time associated with the UAVs.

To achieve the forest monitoring and fire detection with a team of UAVs-WMRs, the following scenario presented in Figure 3.6 is proposed:

i. A team of M WMRs moving within a leader-follower approach will transport the team of N UAVs from the central ground station to their assigned search area, $M = N$.

ii. The UAV team takes-off and start the search and coverage mission. The team is also moving in a leader-follower scheme.

iii. Once a fire is detected by i^{th} UAV, it will send a fire alarm to the remaining UAVs team members and the leader WMR. Afterward, the sensory data will be sent from all UAVs to the leader WMR assigned as the local mobile ground station. Based on these data, the leader WMR controller will re-plan the UAVs reference trajectory and the new formation reconfiguration based on calculating the fire spread model. Eventually, the leader WMR will send the new information to the UAV leader.

iv. The UAV team reconfigures its formation according to the new situation, following the fire perimeter to detect, monitor, and provide updated on-line information about the fire spread, area burnt and still burning.

3.6 Simulation Results

To verify the proposed control strategies presented in this chapter, a team WMRs are used in simulation to validate the effectiveness of the proposed control algorithm. All the values adopted for the control system parameters are shown in Table 3.1.

The weighting matrices Q and R of MPC are selected as:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0.8 \end{bmatrix}, \quad R = I_{2\times 2} \times 0.1$$

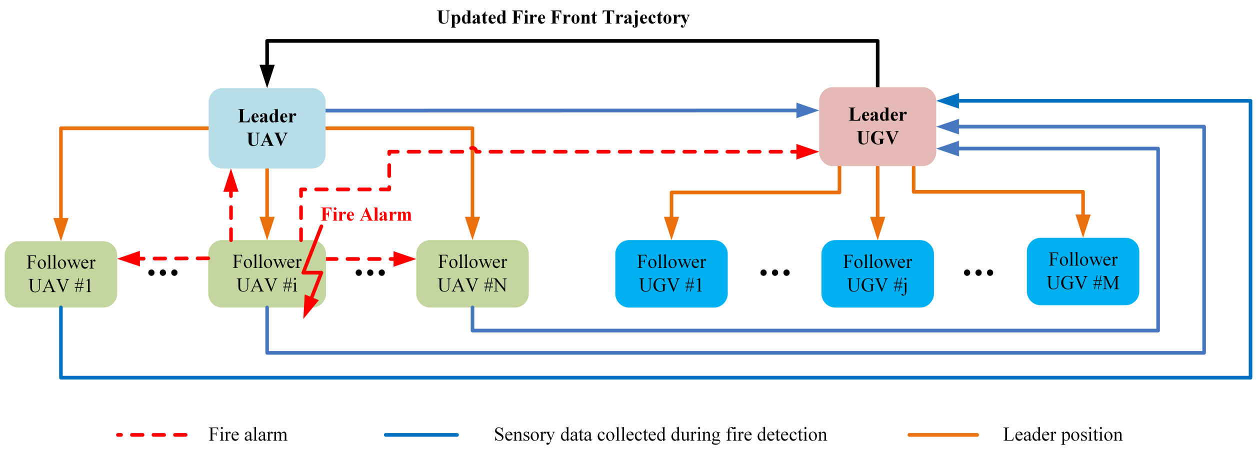


Figure 3.6: Cooperative forest monitoring and fire detection block diagram

Table 3.1: Formation controller parameters

Parameter	Value	Parameter	Value
p	5	c	2
T_s	0.05 sec	F_{max}	1.5
d_{max}	1.5 m	d_{min}	0.5 m
s	3	I	0.15 N.s^2/rad
B	1.3 N.s/rad	K	1.2 N/m

Different cases have been carried out in the simulation to prove the robustness of the proposed controller. These cases are:

- Single robot trajectory tracking;
- Cooperative control in obstacle-free environment;
- Cooperative control in a cluttered environment when the team facing static and moving obstacles; and
- Cooperative control of UAVs - WMRs for forest monitoring and fire detection mission.

3.6.1 Case 1: Single Robot Trajectory Tracking

In this case, single robot tracks an ∞-shaped trajectory. The robot initial position is $(x_o, y_o) = (1, 0.7)$. The simulation time is 30 sec, and the reference trajectory defined by:

$$x_r(t) = 1.1 + 0.7 \sin\left(\frac{2\pi}{30}\right)t \tag{3.44}$$

$$y_r(t) = 0.9 + 0.7 \sin\left(\frac{4\pi}{30}\right)t \tag{3.45}$$

Simulation results show the effectiveness of the proposed control algorithm. Figure 3.7 shows that the robot can track the reference trajectory, while Figure 3.8 shows that the proposed control algorithm converge the actual position and orientation to the reference one. Also, it can be observed from Figure 3.9 that the objective function converges to zero.

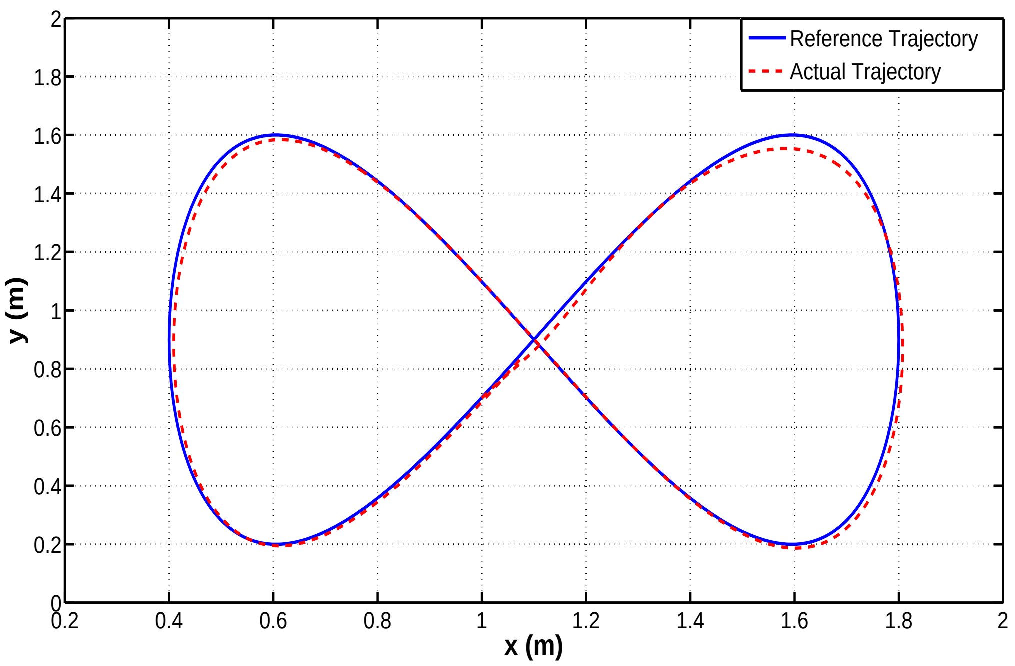


Figure 3.7: Comparison between actual and reference trajectories during simulation

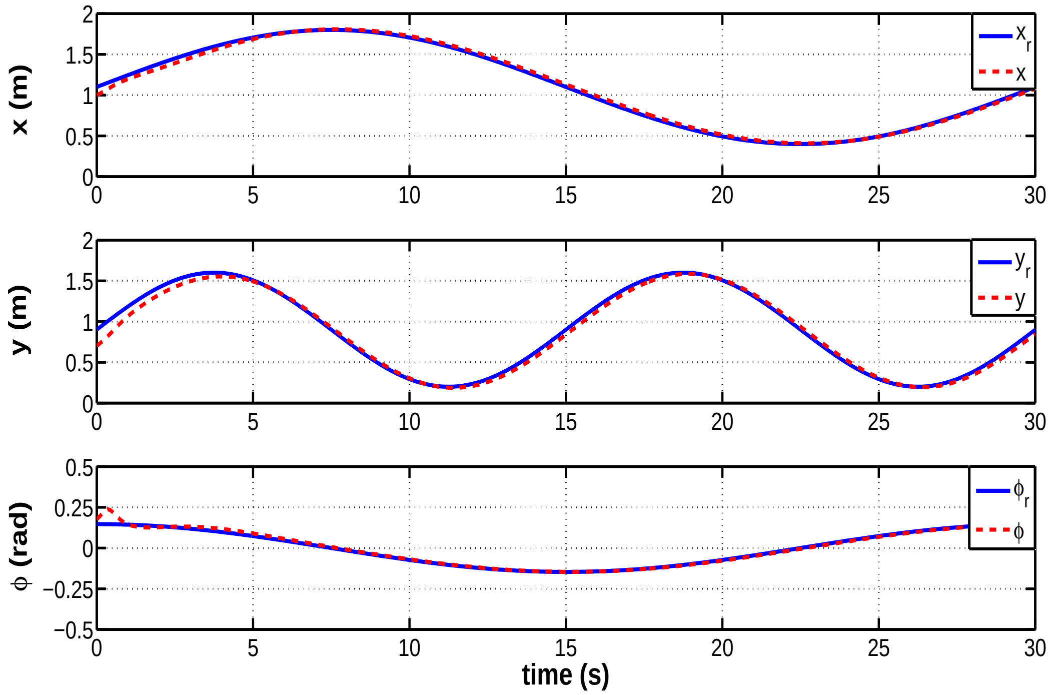


Figure 3.8: Comparison between actual and reference states during simulation

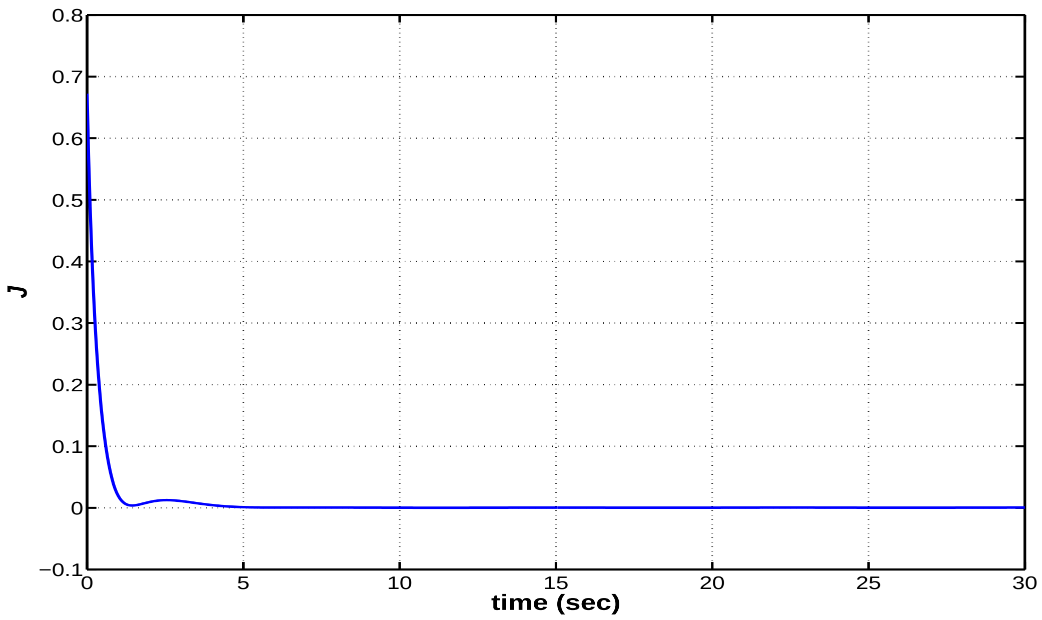


Figure 3.9: Cost function convergence during simulation

3.6.2 Case 2: Cooperative Control in an Obstacle-Free Environment

In this case, a team of three WMRs performing a triangular formation. The leader's initial position is $q_l(0) = [1.1, -0.1, 100°]^T$. The leader tracks a reference trajectory defined as:

$$x_r(t) = \cos(0.25t) \tag{3.46}$$

$$x_r(t) = \sin(0.25t) \tag{3.47}$$

The initial positions of the followers are $q_1(0) = [0.8, -0.1, 100°]^T$ and $q_2(0) = [1.35, -0.1, 100°]^T$, respectively. The desired formation with respect to the leader is 0.5 m and 135° for the first follower, and 0.5 m and 225° for the second one. The simulation time is 23 sec.

Figures 3.10, 3.11, and 3.12 demonstrate the simulation results. From these results, it is clear that the team of WMRs are performing the desired formation. In Figure 3.10, the three WMRs are performing a triangular formation with the desired configuration, while Figures 3.11 and 3.12 show that the followers converge to the desired value of formation distance l_d and angle ϕ_d, respectively.

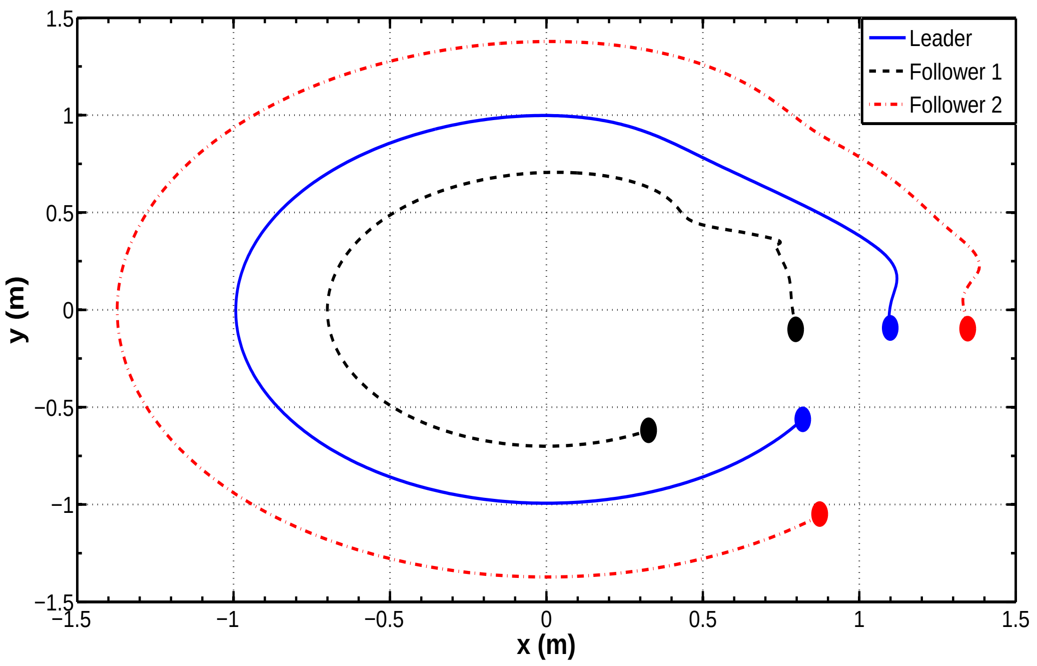


Figure 3.10: Triangular formation of a team of three WMRs during simulation

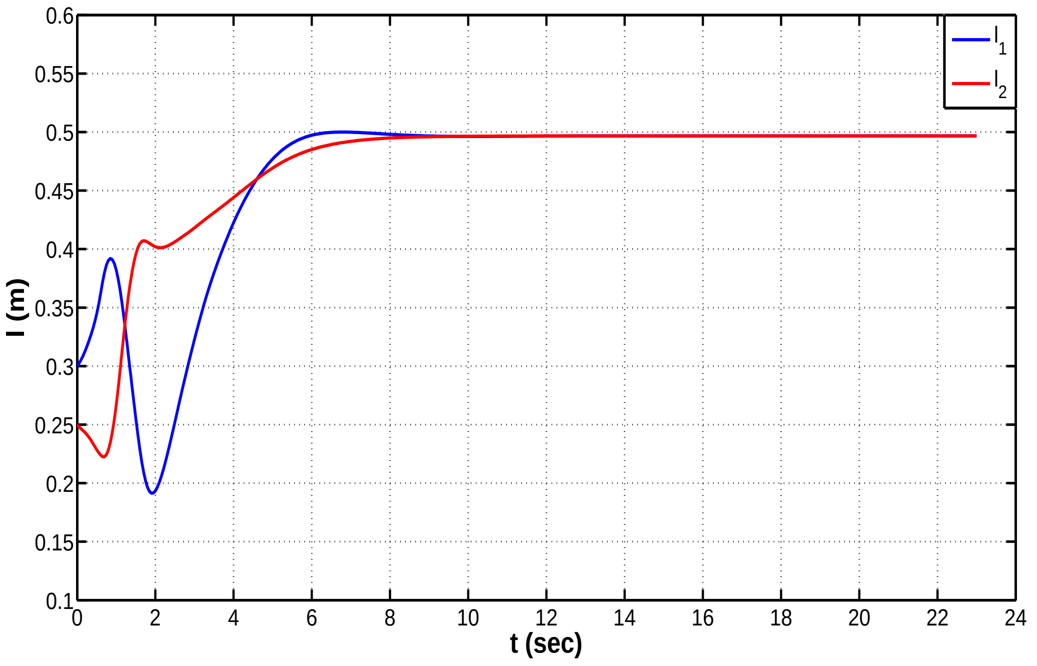


Figure 3.11: Desired formation distances between the leader and the followers during simulation

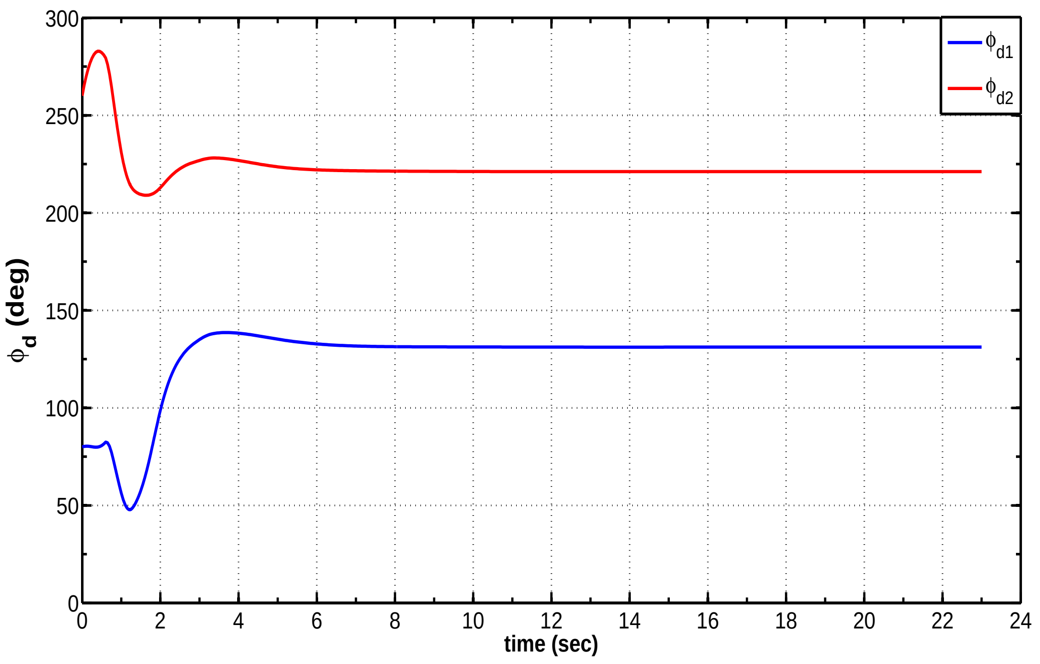


Figure 3.12: Desired formation angles between the leader and the followers during simulation

3.6.3 Case 3: A Team of Cooperative WMRs Facing Static Obstacles

In this case, a team of three WMRs performing a triangular formation should navigate in a partially structured environment containing ten static obstacles, each one is 1 m $\times$ 1 m. The leader's initial position is $q_l(0) = [1, -1, \pi/3]^T$. The first follower's initial position is $q_1(0) = [0, -6, \pi/3]^T$, and its desired formation with respect to the leader is 6.5 m and 140°. While the second follower's initial position is $q_2(0) = [-5, 2, \pi/3]^T$ and its desired formation with respect to the leader is 6.5 m and 220°. The simulation time is 80 sec.

Simulation results are presented in Figures 3.13 to 3.15. it is clear that the team of WMRs are performing the desired formation and avoiding the obstacles that may face during the mission. In Figure 3.13, the three WMRs are performing a triangular formation with the desired configuration avoiding the obstacles located in the environment. Obstacles 1 and 2 are located 1.5 m away from the leader and the first and second follower respectively (which is equal to d_{max}), so neither the leader nor the followers are affected by these obstacles. Obstacles 3, 4, 5 and 6 are near the first and second followers, so the repulsive forces generated cause a change in both the linear and angular velocities of the robots to avoid the obstacles. Obstacles 7, 8, 9 and 10 are far away from the robots, so no change in the robots movement.

Figure 3.14 shows the linear and angular velocities of the robots. As shown, the effect of v_c and ω_c appears for the first and second followers at $t = 28$ and 18 sec respectively in order to avoid the detected obstacles. Moreover, Figure 3.15 shows that the constraint presented in Eq. (3.40) is achieved, except during the obstacle avoidance. Once the robots passed over the obstacles, the error in the desired formation converges to zero.

3.6.4 Case 4: A Team of Cooperative WMRs Facing Moving Obstacles

In this case, a team of three WMRs performing a triangular formation facing moving obstacles. It is assumed that there are two obstacles moving toward the followers. As the first case, each obstacle is 1 m × 1 m. The Cartesian equation presenting the path of the obstacle facing the first follower is defined as:

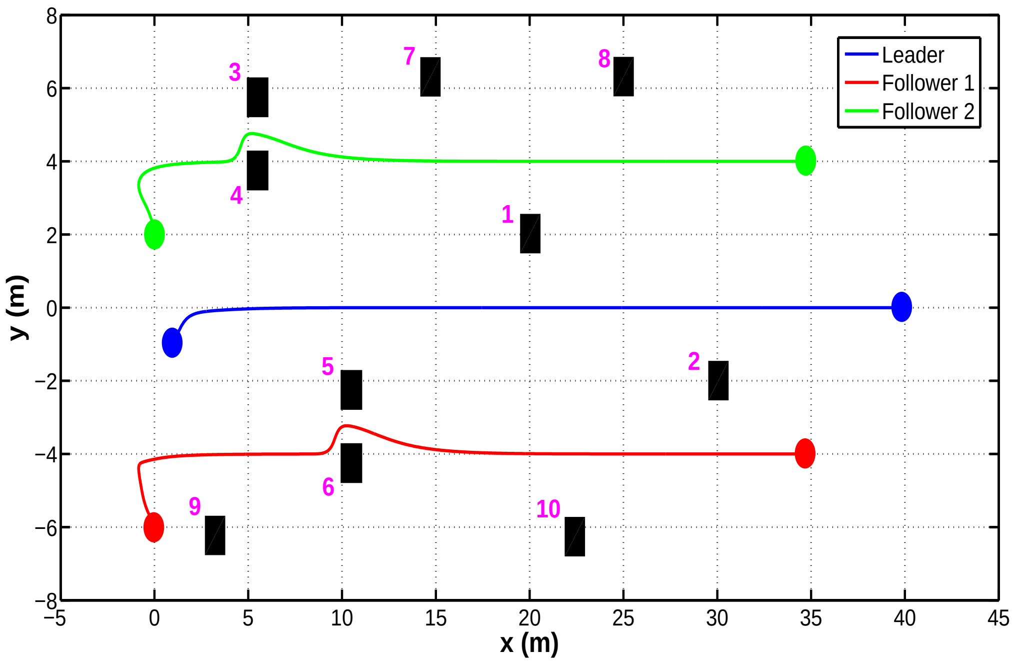


Figure 3.13: Desired formation of a team of two WMRs in the presence of static obstacles during simulation

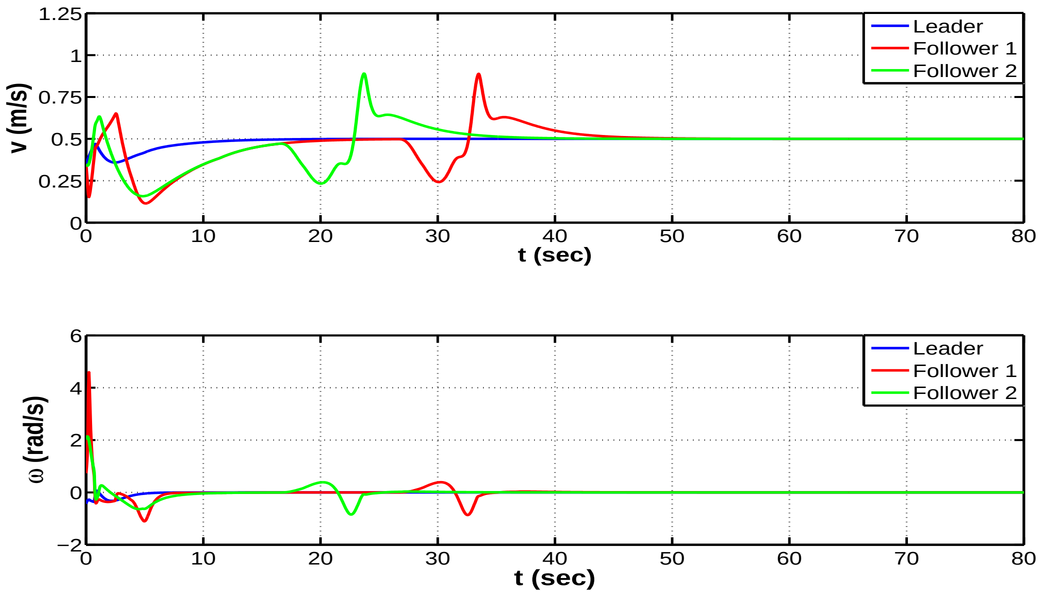


Figure 3.14: The effect of the virtual force on the linear and angular velocities of the robots when facing static obstacles during simulation

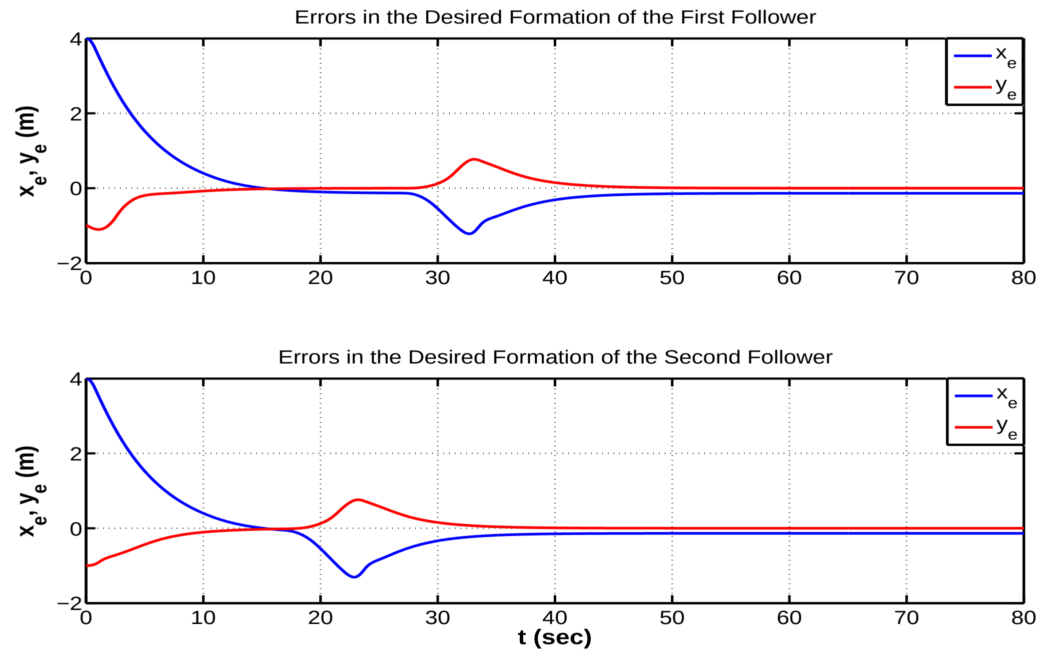


Figure 3.15: Errors in the desired formation of the followers when facing static obstacles during simulation

$$\begin{aligned} x_{obs1} &= 32 - 0.2t, \\ y_{obs1} &= 6 - 0.2t \end{aligned} \tag{3.48}$$

and for the obstacle facing the second follower:

$$\begin{aligned} x_{obs2} &= 18 - 0.2t, \\ y_{obs2} &= 10 - 0.2t \end{aligned} \tag{3.49}$$

Figures 3.16 to 3.18 present the simulation results. It is clear that the team of WMRs are performing the desired formation and avoiding the moving obstacles as shown in Figure 3.16. In Figure 3.17, the linear and angular velocities of the robots are illustrated. As shown, the effect of v_c and ω_c appears for the first and second followers at $t = 50$ and 30 sec, respectively in order to avoid the moving obstacles. Furthermore, Figure 3.18 presents the distance between each robot and its nearest obstacle. Once each robot detects the obstacle and the distance d between them becomes less than the maximum allowable distance d_{max}, each robot tries to avoid the obstacle and the distance between them does not exceed the minimum acceptable distance d_{min} as shown in the figure.

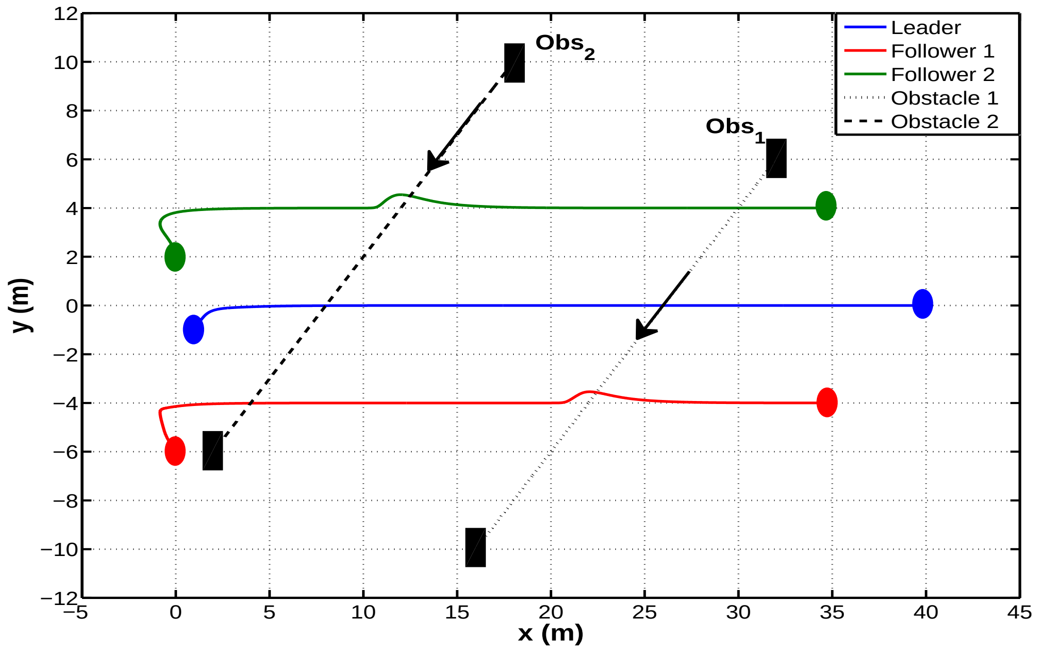


Figure 3.16: Triangular formation within moving obstacles

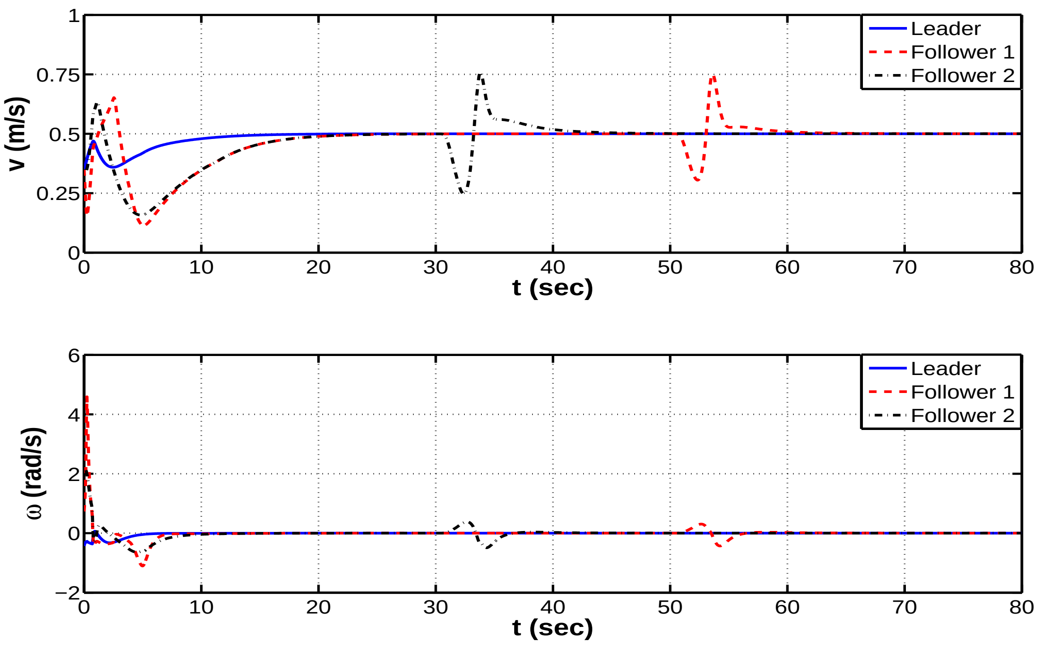


Figure 3.17: Linear and angular velocities of the robots facing moving obstacles

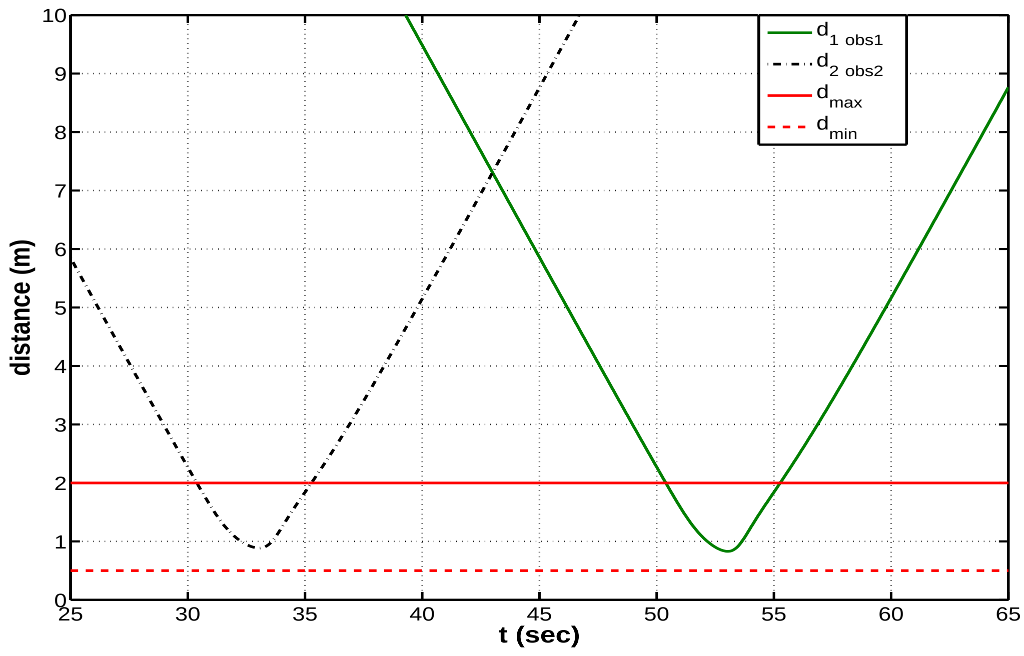


Figure 3.18: Distances between each robot and the nearest obstacle when facing moving obstacles

3.6.5 Case 5: Cooperative Control of UAVs - WMRs for Forest Monitoring and Fire Detection Mission

In this case, a team consisting of three UAVs and three WMRs executes a forest monitoring and fire detection mission. The WMR leader robot's initial position is $q_1(0) = [17.5, -43, \pi/3]^T$. The first follower's initial position is $q_1(0) = [15, -47.33, \pi/4]^T$, and its desired formation with respect to the leader is 5 m and 150°. The second follower's initial position is $q_2(0) = [20, -47, \pi/3]^T$ and its desired formation with respect to the leader is 5 m and 210°. The UAVs desired formation during search and coverage with respect to their leader is 5 m and 150° for the first UAV, and 0.5 m and 210° for the second one. It is assumed that the first follower UAV detects a fire spot after 70 sec of mission starting.

Figure 3.19 illustrates the formation of the team during mission execution. As can be seen, the team of WMRs start the mission by carrying the UAVs until the nearest point (marked with the $\diamond$ marker) to the assigned search area. The UAV team takes-off and start the search mission. The first follower UAV detects a fire spot with coordinates (0,-25,0). The leader WMR plans the reference trajectory that the UAVs must follow according to the fire spread model. The UAV starts to reconfigure its formation to follow the new trajectory at the instant marked with the ($\triangleleft$) black marker. The UAV team surrounds the fire spot following the elliptic trajectory with formation angle 120°.

Figures 3.20 and 3.21 show the effectiveness of the proposed WMRs formation controller. As can be seen, the follower WMRs maintain the desired formation distances and angles between them and the leader, respectively.

3.7 Experimental Results

To validate the performance and robustness of the proposed control algorithms, experimental tests are preformed, The following cases to be presented:

- Single robot trajectory tracking;
- Cooperative control in obstacle-free environment; and
- Cooperative control in a cluttered environment when the team facing static obstacles.

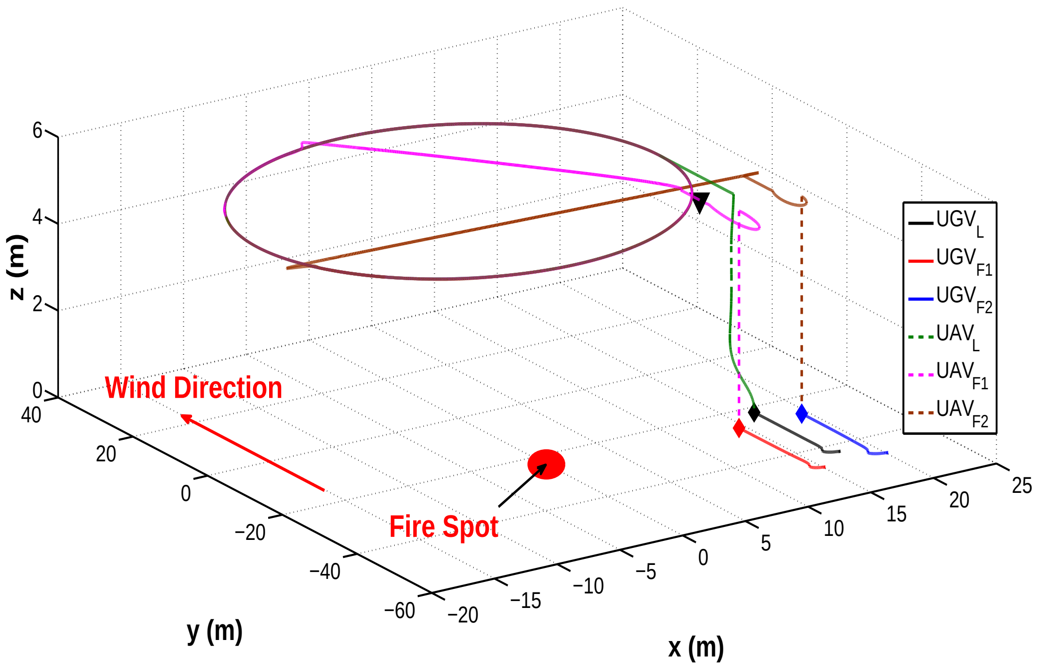


Figure 3.19: Formation of the WMRs-UAVs team during forest monitoring and fire detection mission

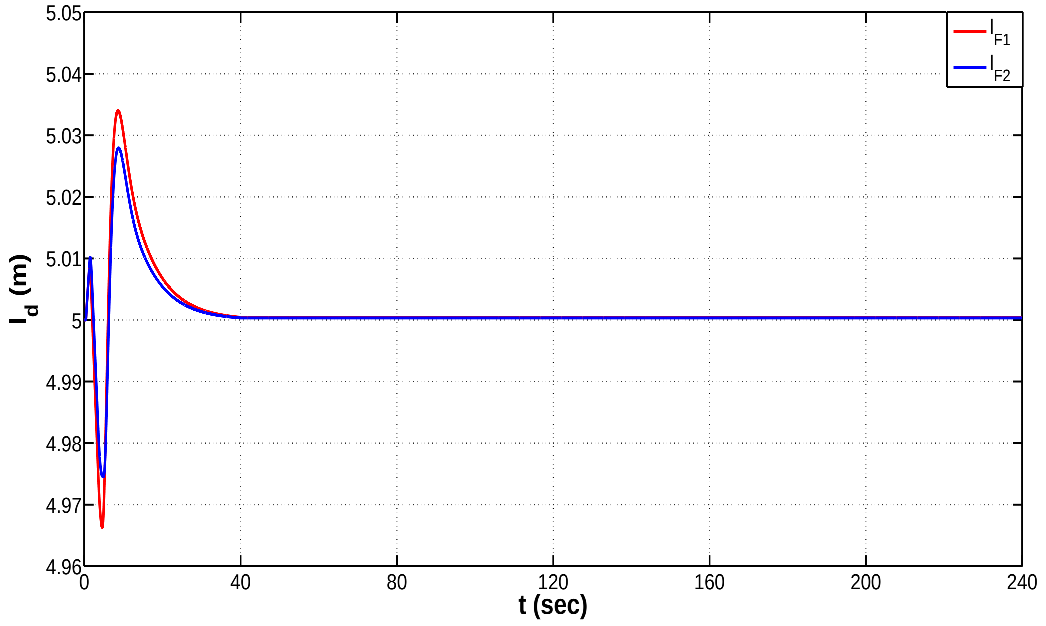


Figure 3.20: Desired formation distances between the leader WMR and their followers during forest monitoring and fire detection mission

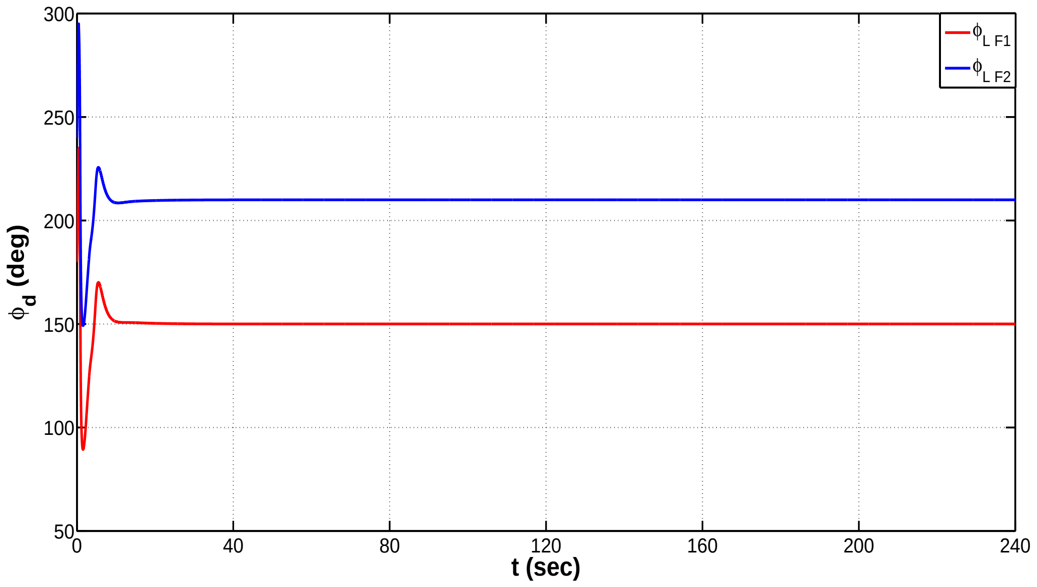


Figure 3.21: Desired formation angles between the leader WMR and their followers during forest monitoring and fire detection mission

3.7.1 Case 1: Single Robot Trajectory Tracking

In this case, two experiments are performed; tracking an ∞-shape, and tracking a square shape trajectories. In the first case, a single robot tracks an ∞-shaped trajectory. The robot initial position is $(x_o, y_o) = (-0.16, -0.2)$, while the reference trajectory defined by:

$$x_r(t) = 0.7 \sin\left(\frac{2\pi}{30}\right) t \tag{3.50}$$

$$x_r(t) = 0.7 \sin\left(\frac{4\pi}{30}\right) t \tag{3.51}$$

In the second experiment, the robot tracks a square shaped trajectory. The robot initial position is $(x_o, y_o) = (-1.03, 1.1)$. The time of both experiments are 30 sec and 40 sec, respectively.

As can be seen in Figures 3.22 to 3.25, the proposed control algorithm exhibits good performance in real-time, and able to compensate the vision system delay, and the noisy data. In Figure 3.22, and Figure 3.24, the robot able to track the reference trajectories, Figure 3.23 shows that robot states converge to the reference states during tracking the ∞-shaped trajectory, while Figure 3.25 illustrates that the error in robot posture converge to zero.

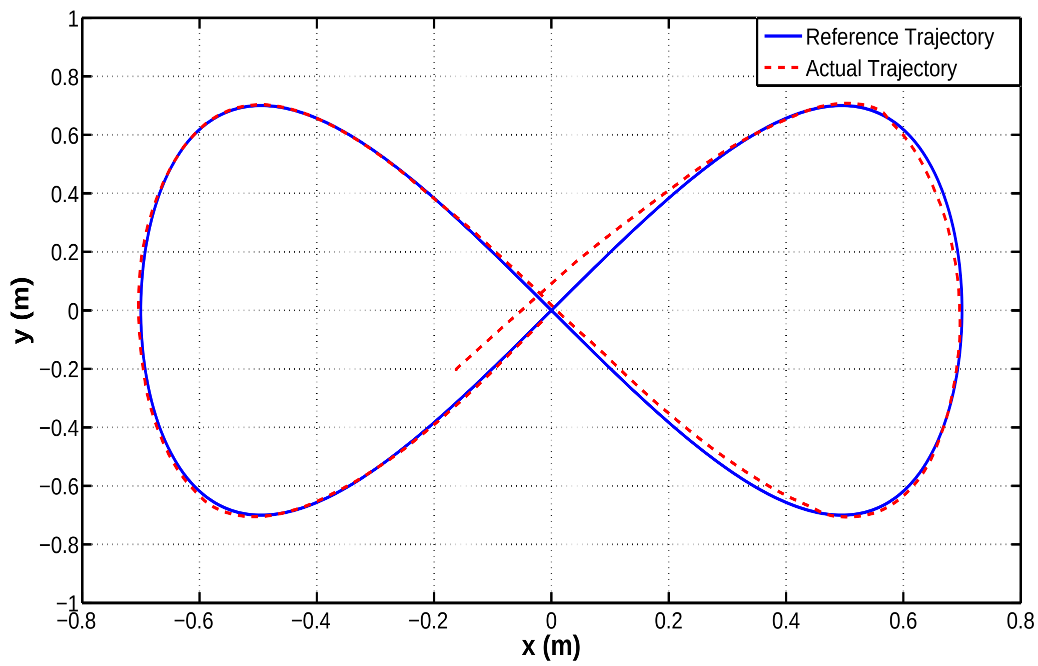


Figure 3.22: Experimental testing of tracking an ∞-shaped trajectory

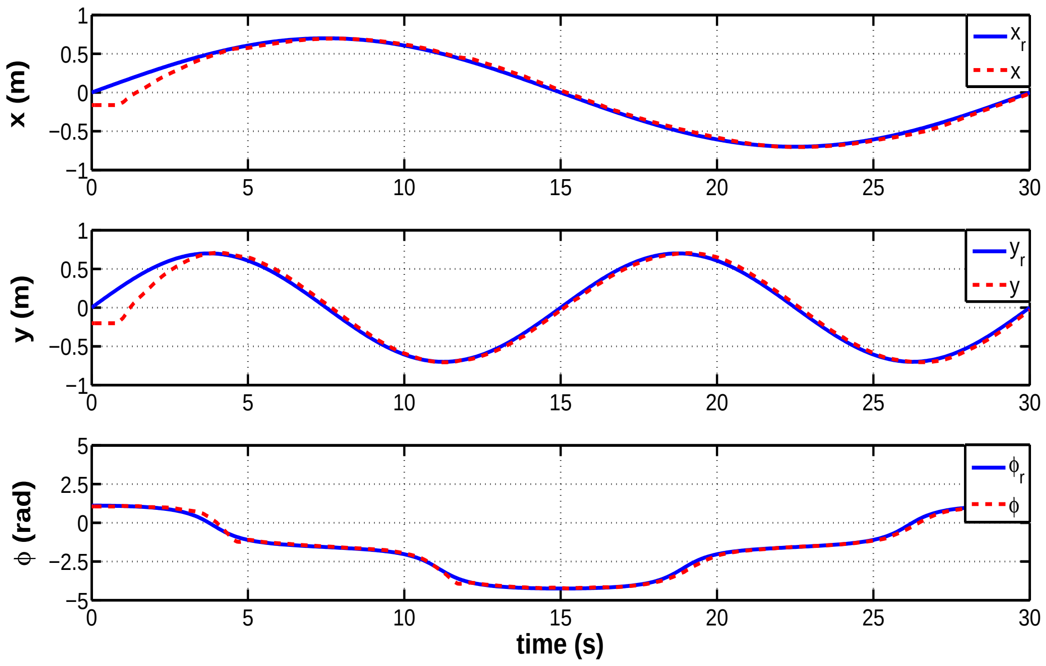


Figure 3.23: Comparison between real and reference states during tracking an ∞-shaped trajectory in real-time experiment

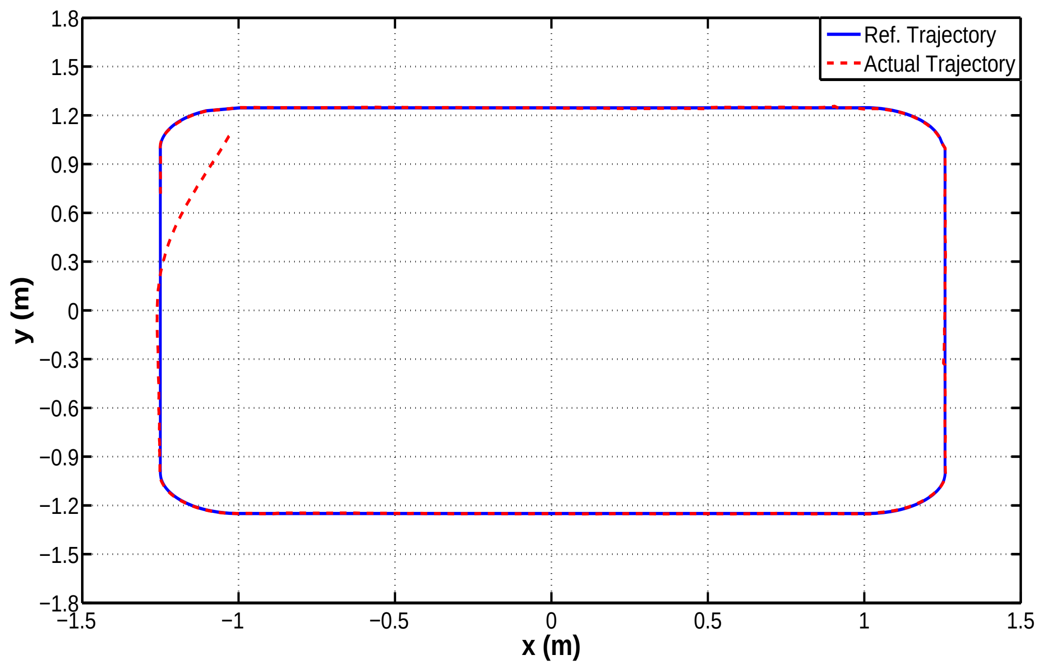


Figure 3.24: Experimental testing of tracking a square shape trajectory

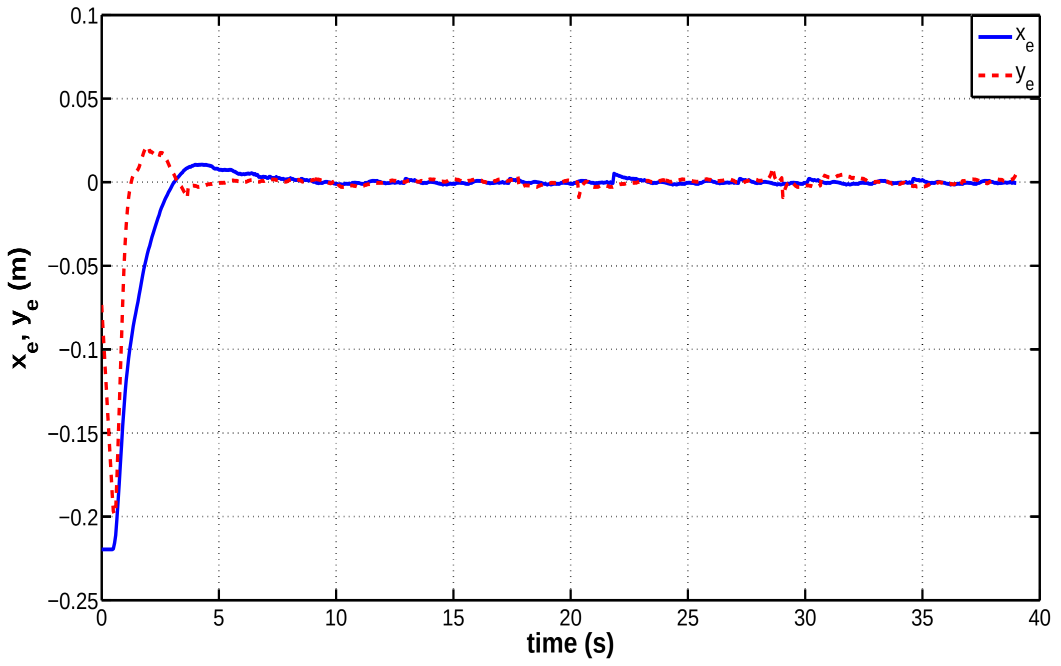


Figure 3.25: Error in robot posture during tracking a square shape trajectory in real-time experiment

3.7.2 Case 2: Cooperative Control in an Obstacle-Free Environment

In this experiment, three WMRs perform a triangular formation. The initial position of the leader is $q_l(0) = [-0.05, -2.05, 100°]^T$. The leader tracks a reference trajectory defined as:

$$x_r(t) = \cos(0.25t) - 0.1 \tag{3.52}$$

$$y_r(t) = \sin(0.25t) - 0.6 \tag{3.53}$$

The initial positions of the followers are $q_1(0) = [-0.55, -2.8, 100°]^T$ and $q_2(0) = [-0.75, -1.32, 100°]^T$, respectively. The desired formation with respect to the leader is 0.75 m and 225° for the first follower, and 0.75 m and 135° for the second one.

Figures 3.26 to 3.28 present the experimental results of this case. From these results, it is clear that the team of WMRs are performing the desired formation. In Figure 3.26, the followers maintain the desired configuration with respect to the leader, while Figures 3.27 and 3.28 show that the formation configuration converge to the desired values l_d and angle ϕ_d, respectively, achieving the constraint presented in (3.40).

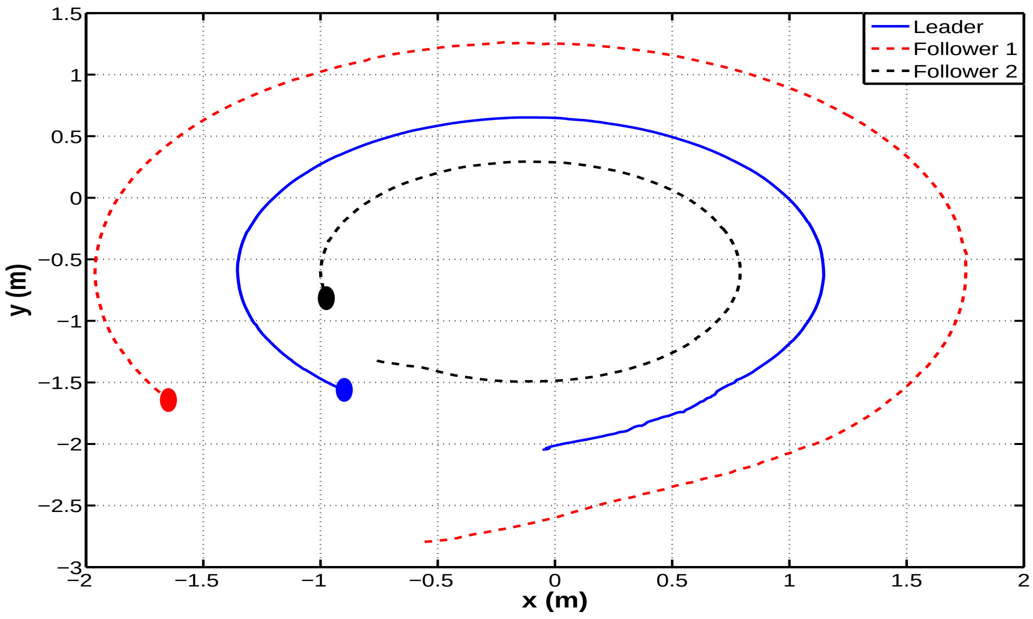


Figure 3.26: Triangular formation of a team of WMRs during experiment

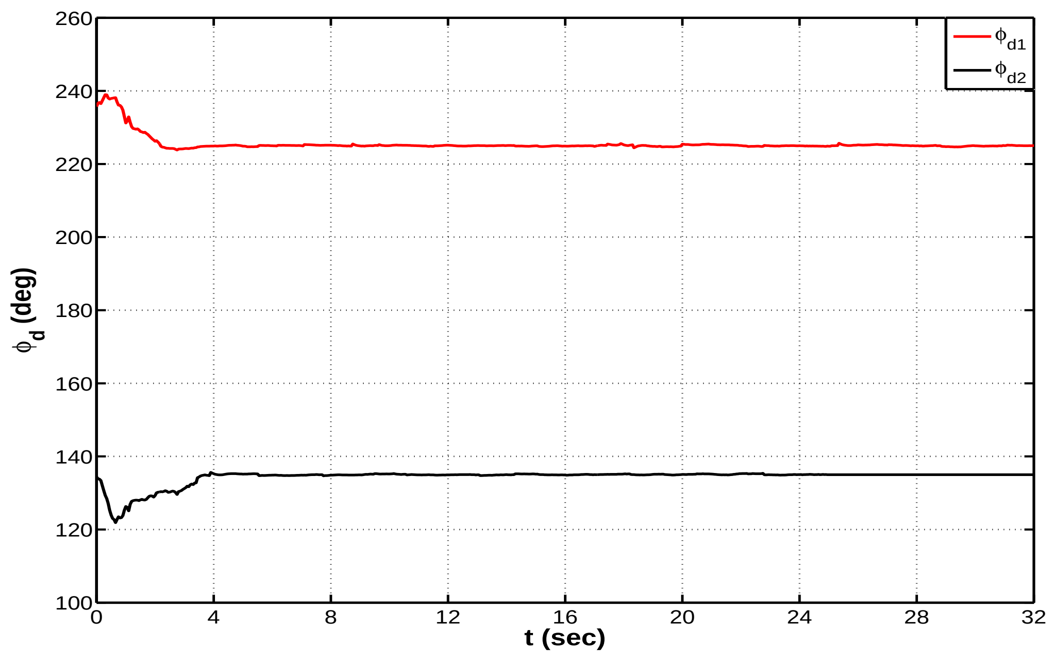


Figure 3.27: Desired formation distances between the leader and the followers during experiment

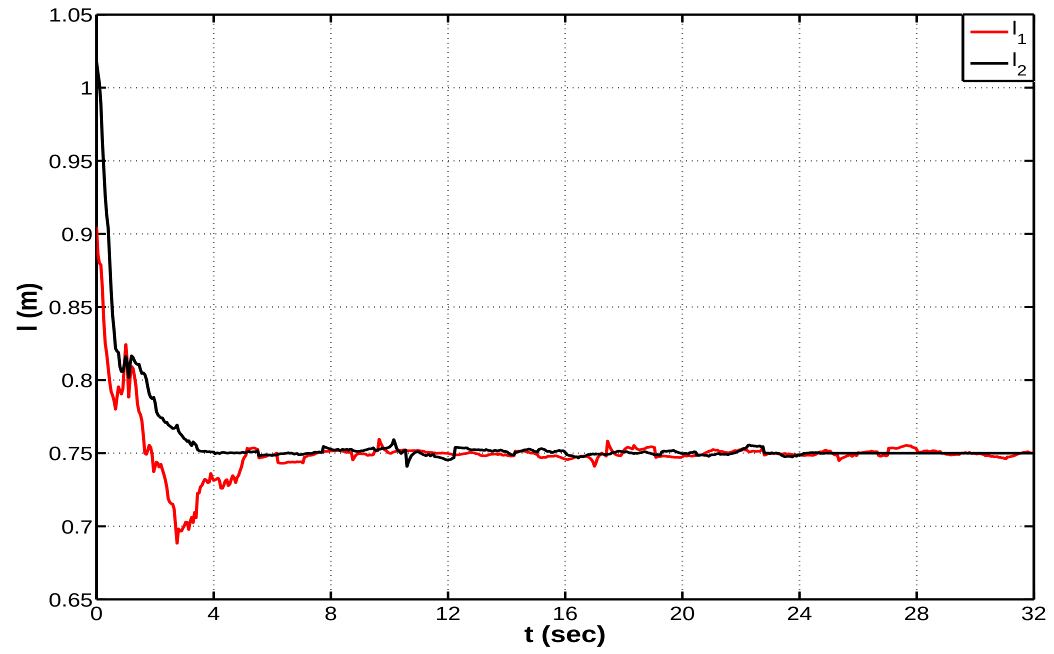


Figure 3.28: Desired formation angles between the leader and the followers during experiment

3.7.3 Case 3: Cooperative Control in a Cluttered Environment

In this case a team of WMRs consists of a leader and a follower performing a desired formation in the presence of obstacles. The leader's initial position is $q_l(0) = [0.05, -1.8, 20°]^T$, while the follower initial position $q_1(0) = [-0.8, -0.92, 60°]^T$ and the desired formation configuration is 1 m and 135° with respect to the leader.

Figures 3.29 to 3.31 present the experimental results. From these results of this case, it is clear that the team of WMRs are performing the desired formation and the follower avoid the obstacle that it face during the formation mission. In Figure 3.29, the follower maintains the desired configuration with respect to the leader except during avoiding the obstacle, once the follower avoid the obstacle, it continue following the leader maintaining the desired formation configuration. Figure 3.30 shows the effect of the virtual force on the linear and angular velocities of the follower. As shown, during the time interval $t \in [16.65, 21.65]$ sec, the virtual repulsive force F to be generated, according to this force v_c and ω_c are calculated and added to the original v and ω respectively. Furthermore, Figure 3.31 shows that the constraint presented in (3.40) is achieved, except during the obstacle avoidance.

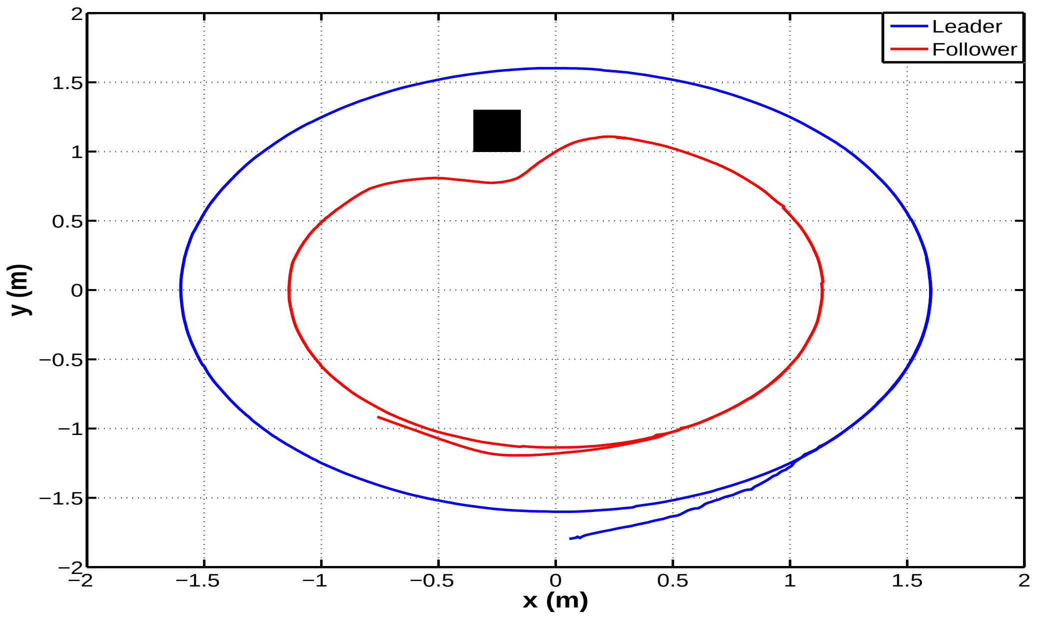


Figure 3.29: Experimental result when a team of two WMRs performing a desired formation in the presence of obstacle

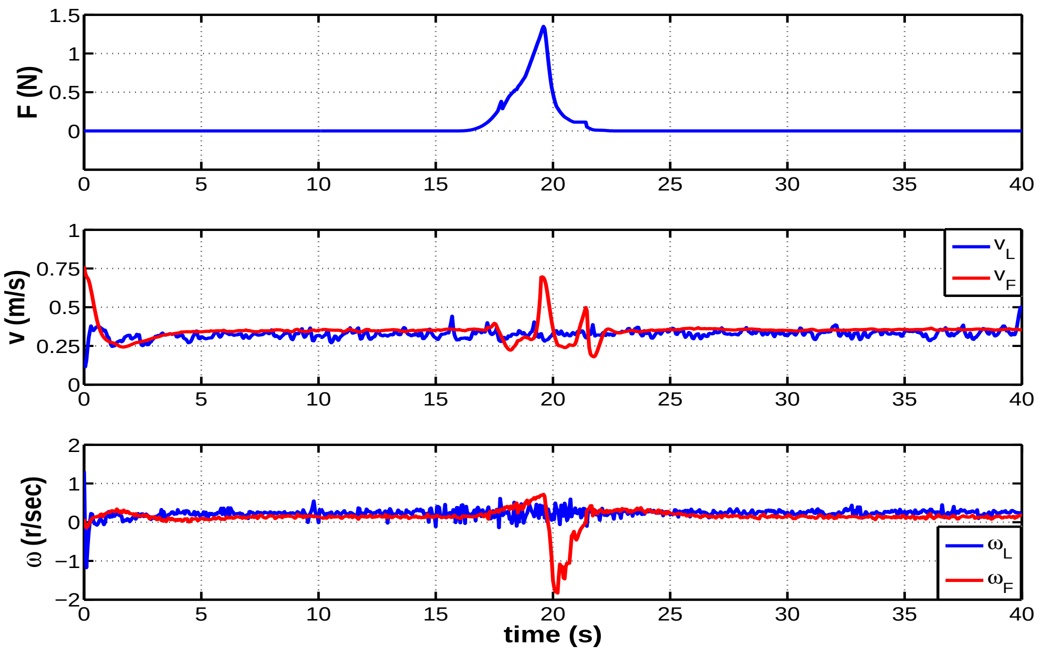


Figure 3.30: The effect of the virtual force on the linear and angular velocities of the robots during experiment

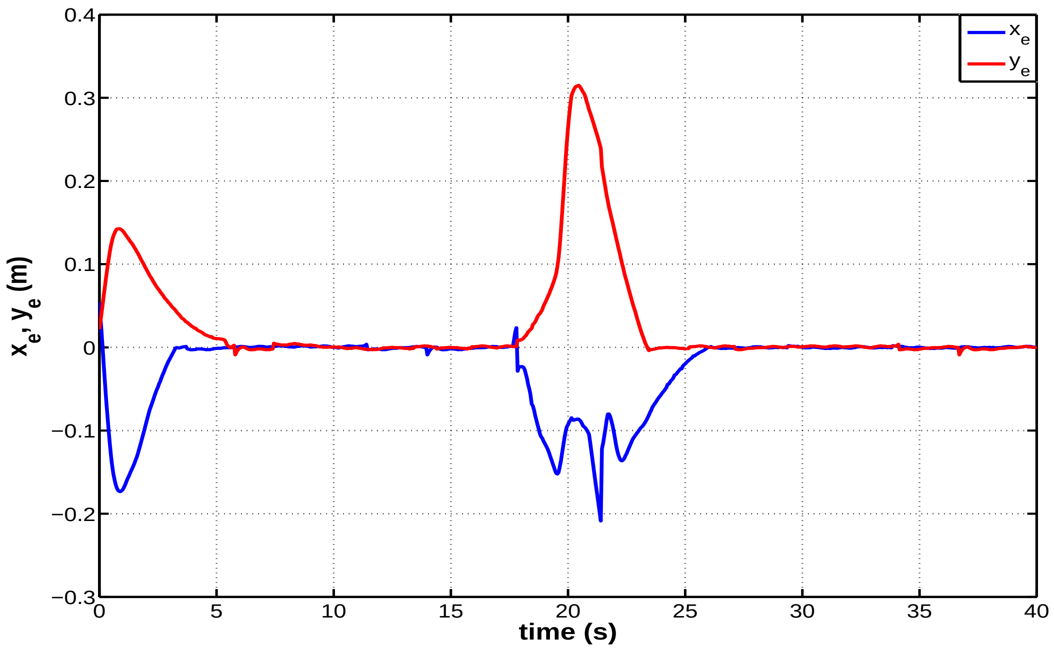
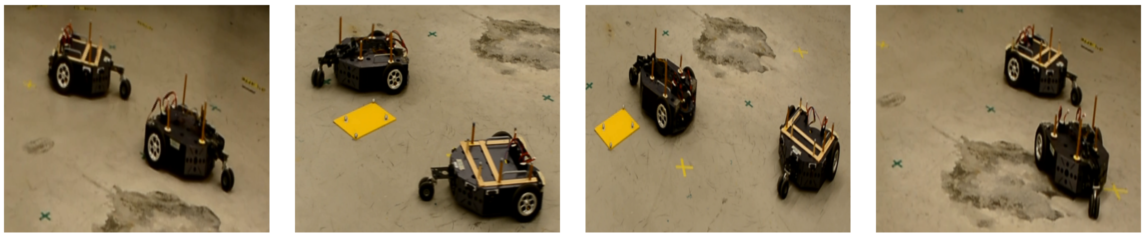


Figure 3.31: Errors in the desired formation of the follower during experiment

Figure 3.32 shows snapshots of the real experiment performed in the laboratory. When $t = 6$ sec, the robots perform the desired formation configuration. When $t = 17$ sec, the follower detects the obstacle and start avoiding it. At $t = 22$ sec, the follower completely avoid the obstacle and continue the mission. When $t = 36$ sec, the leader and the follower continue the planned mission.



(a) $t = 6$ sec (b) $t = 17$ sec (c) $t = 22$ sec (d) $t = 36$ sec

Figure 3.32: Snapshots of the obstacle avoidance experiment

Chapter 4

FTCC of Multiple WMRs Under Actuator Faults

This chapter presents the development of an FTCC algorithm for a team of WMRs in the presence of actuator faults. The main purpose of the FTCC algorithm is to:

- Re-assign the formation mission on the healthy members if one or more robots subject to severe faults and cannot complete the mission;
- Let the team continue the mission with degraded performance if one or more robots subject to a fault but still able to complete the mission; and
- Avoid the potential collision between the healthy robots and the faulty one if it gets out from the formation due to severe fault occurrence.

As a result, the FTCC algorithm consists of:

- The FDD scheme in order to detect and diagnose the actuator fault;
- The decision-making algorithm in which the mission is either re-coordinated or re-assigned based on the degree of severity of the occurred fault; and
- The collision avoidance algorithm.

In case of severe fault occurrence, there are two main challenges; how to generate the new formation configuration, and how to reconfigure the formation to the new one. As mentioned in Chapter 2, during formation reconfiguration, sudden changes have occurred to the robots' control inputs. If these inputs are not bounded, the robots behavior become unstable, the formation may be broken, and the robots may collide. Since MPC is adopted as the WMRs' controllers as presented in Chapter 3, the second problem can be solved considering the input constraints during reconfiguration. Therefore, in this Chapter, the first problem is considered based on two techniques: **i)** the Graph Theory; and **ii)** formulating the FTCC problem as an optimal assignment problem, where a Hungarian algorithm is applied to solve it. To obtain the information of the actuator faults, the TSKF is used, where the faults are modeled as losses in the effectiveness of the robots' driving motors. The advantage of the TSKF is to simultaneously estimate the states and fault parameters for the FDD purpose.

This chapter is organized as follows. First, the design of FDD scheme is presented. Next, the decision-making algorithm including both task re-assignment and motion re-coordination is illustrated. Finally, simulation and experimental results are presented to validate the effectiveness of the proposed algorithms.

Remark 4.1. *The obstacle avoidance algorithm presented in Section 3.4 is applied here to achieve the collision avoidance with the faulty robots.*

4.1 Fault Detection and Diagnosis Scheme (FDD)

For safe, reliable and secured mission execution, WMRs should have an FDD algorithm capable of monitoring actuators' health without requiring any sensors. This is very important since better knowledge of the fault location, type, and amplitude greatly helps in minimizing the fault effects on the system behavior (Chamseddine et al., 2014).

As a result, the main purpose of the FDD algorithm is to:

- Detect the abnormal behaviors of a process due to a component failure;
- Eventually isolate the exact location of the failed component; and
- Identify the fault type and its severity.

As mentioned in Section 2.3.1, differentially-driven WMRs are equipped with two motors. According to the value of the optimum control inputs given from MPC, the angular velocities of the driving motors can be calculated according to Eq. (2.6). The angular velocities are converted to the corresponding applied voltage, then the motors are driven corresponding to this applied voltage.

The basic idea of the FDD scheme is to estimate the loss of effectiveness of the motors based on the difference between the actual values of the motors' angular velocities and the theoretical ones (obtained form the robot controller) as shown in Figure 4.1. Therefore, if there is a difference between them, there is a loss of effectiveness of the robot actuator, and the value of this loss of effectiveness should be estimated. It is modeled as a random bias, and the TSKF is used to detect and estimate the value of this bias.

Assumption 4.1. *Each robot in the formation has its own* FDD *scheme. Therefore, each robot can detect the occurred faults and estimate their loss of effectiveness*

4.1.1 Robot Actuator Modeling

DC motors are widely used in robotic applications and are the main type of actuators used in mobile robots. Since the TSKF is considered as a model-based FDD, modeling of the DC motor is needed first.

In general, the torque T generated by a DC motor is proportional to the armature current and the strength of the magnetic field. Assuming that the magnetic field is constant and, therefore, that the motor torque is proportional only to the armature current i by the torque constant k_t as:

$$T = k_t i. \tag{4.1}$$

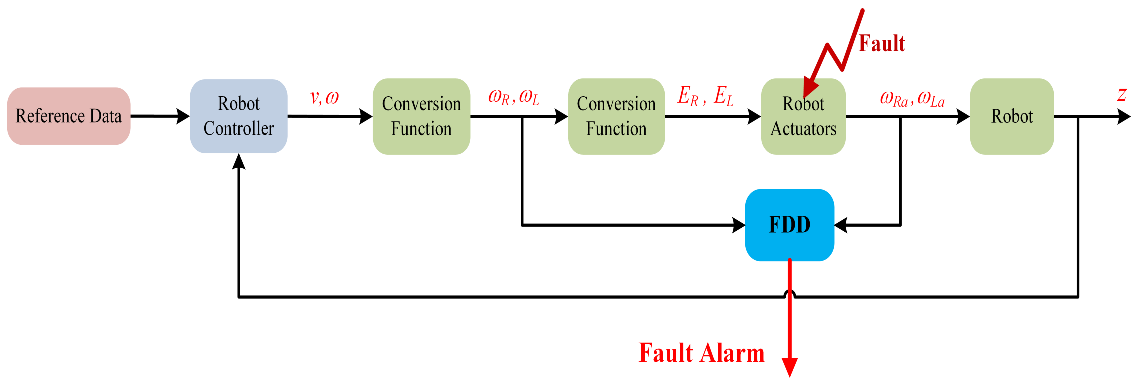


Figure 4.1: Control system block diagram for each robot including the FDD unit

The back emf, e, is proportional to the angular velocity of the shaft by a back emf constant k_e

$$e = k_e\omega. \tag{4.2}$$

Usually, the motor torque and back emf constants are equal, then let $k_t = k_e = K_c$. Therefore, the governing equations of the driving motor are:

$$I_m\frac{d\omega}{dt} + b\omega = K_c i, \tag{4.3}$$

$$L_a\frac{di}{dt} + R_a i = E - K_c\omega, \tag{4.4}$$

where I_m is the moment of inertia, b is the motor viscous friction constant, L_a is the inductance, R_a is the armature resistance, and E is the applied voltage.

The governing equations above can be expressed in state-space representation by choosing the rotational speed and electric current as two state variables. The applied voltage is treated as the input and the rotational speed is chosen as the output, then the governing equations can be written as:

$$\begin{gathered}\begin{bmatrix}\dot{\omega}\\ \dot{i}\end{bmatrix} = \begin{bmatrix}-\frac{b}{I_m} & \frac{K_c}{I_m}\\ -\frac{K_c}{L_a} & -\frac{R_a}{L_a}\end{bmatrix}\begin{bmatrix}\omega\\ i\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{L_a}\end{bmatrix}E,\\ \omega = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\omega\\ i\end{bmatrix}\end{gathered} \tag{4.5}$$

The state-space model in (4.5) can be extended for the two driving motors as follows:

$$\begin{gathered}\begin{bmatrix}\dot{\omega}_R\\ \dot{i}_R\\ \dot{\omega}_L\\ \dot{i}_L\end{bmatrix} = \begin{bmatrix}-\frac{b}{I_m} & \frac{K_c}{I_m} & 0 & 0\\ -\frac{K_c}{L_a} & -\frac{R_a}{L_a} & 0 & 0\\ 0 & 0 & -\frac{b}{I_m} & \frac{K_c}{I_m}\\ 0 & 0 & -\frac{K_c}{L_a} & -\frac{R_a}{L_a}\end{bmatrix}\begin{bmatrix}\omega_R\\ i_R\\ \omega_L\\ i_L\end{bmatrix} + \begin{bmatrix}0 & 0\\ \frac{1}{L_a} & 0\\ 0 & 0\\ 0 & \frac{1}{L_a}\end{bmatrix}\begin{bmatrix}E_R\\ E_L\end{bmatrix},\\ \begin{bmatrix}\omega_R\\ \omega_L\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\omega_R\\ i_R\\ \omega_L\\ i_L\end{bmatrix}\end{gathered} \tag{4.6}$$

where the subscripts R and L denote for the right and left motors respectively. The DC motor dynamics is shown in Figure 4.2.

4.1.2 The Two-Stage Kalman Filter (TSKF) Design

In 1960, Rudolf E. Kálmán introduced his approach to linear filtering based on the method of minimum variance (Kalman, 1960). Compared to existing filtering techniques at that time, the Kalman filter, though usually more computationally intense, offered performance improvements and ease of implementation on a digital computer due to its recursive formulation. Moreover, the Kalman filter processes all available measurement data or information that can be provided to it, regardless of their precision, on the basis of their stochastic descriptions, in order to generate an overall best estimate of the parameter considered (Ducard, 2007).

Although a mathematical model for the robot actuators has been developed to describe their behavior, but this model will never be perfect, leaving many effects unmodeled (such as the effect of the friction forces and slippage between the robots' wheels and the ground). Furthermore, several parameters of the model will not be known exactly, and the sensor measurement data will be corrupted by noise and biases. For all those reasons, Kalman filtering techniques are good choice to consider such system dynamics and measurement noises, errors, and uncertainties.

Compared to the regular Kalman filter, the TSKF has the advantage of simultaneously estimate both states and fault parameters, for the purpose of fault detection, isolation, and identification as well as providing full-state estimation for state feedback-based controllers when state vector is not available through measurements (Chamseddine et al., 2014).

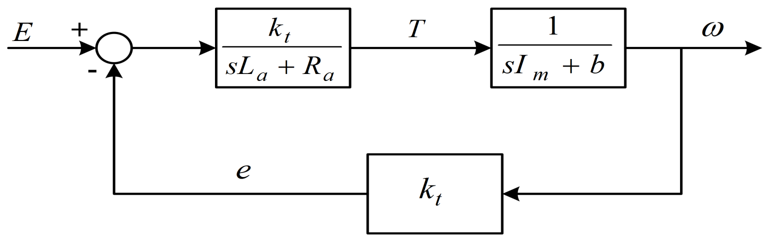


Figure 4.2: DC motor block diagram

The first step to develop the TSKF is representing the mathematical model of the robot actuators described in Eq. (4.6) in the following discrete formula:

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k u_k + w_k^x \\ y_{k+1} &= C_k x_{x+1} + \upsilon_{k+1} \end{aligned} \tag{4.7}$$

where $x_k \in \mathbb{R}^{n_d}$, $u_k \in \mathbb{R}^{m_d}$, and $y_{k+1} \in \mathbb{R}^{p_d}$. n_d, m_d, and p_d are the state, control, and output variables respectively. $n_d = 4$, $m_d = 2$, and $p_d = 2$. w_k^x and υ_{k+1} are white noise sequences of uncorrelated Gaussian random vectors with zero means and covariance matrices Q_k^x and R_k, respectively.

When applying the Kalman filtering techniques to any process, the accurate model of the process dynamics is required. However, in practical cases, the system dynamics may be affected by a constant bias. If this bias is not integrated with the model, then a performance degradation of the filter may be occurred. Considering a bias vector $w_k^y \in \mathbb{R}^{P_r}$, then the model presented in (4.7) can be written as:

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k u_k + F w_k^y + w_k^x \\ w_{k+1}^y &= w_k^y + \upsilon_k^w \\ y_{k+1} &= C_k x_{x+1} + \upsilon_{k+1} \end{aligned} \tag{4.8}$$

where υ^w is an uncorrelated Gaussian random vector with zero mean and covariance matrix Q_k^w. To estimate the bias vector w_k^y, it should be augmented into the state vector to make an augmented state vector which is estimated by using the augmented state Kalman filter (ASKF). The drawback of the ASKF is the computational burden (Chamseddine et al., 2014), since the dimension of the augmented state vector is $n_d + P_r$. As a result, Keller and Darouach (1997) try to avoid this drawback by using two parallel reduced-order filters which optimally implement the augmented state filter. The proposed algorithm is called TSKF.

In the field of FDD, the effectiveness of actuators is estimated as the augmented random bias vector. For a team of N robots, When i^{th} actuator fails in the robot j, then the control inputs are represented by:

$$u_F^{j_i} = u^{j_i}(1 - \gamma_k^{j_i}) \quad 0 \leq \gamma_k^{j_i} \leq 1, \quad i = 1, \ldots, m_d, \quad j = 1, \ldots, N \tag{4.9}$$

where $\gamma_k^{j_i}$ represents the loss of control effectiveness in i^{th} actuator of the j^{th} robot. If $\gamma_k^{j_i}$ $= 0$, then control input is normal; if $\gamma_k^{j_i} = 1$, then u^{j_i} is outage; and if $0 < \gamma_k^{j_i} < 1$, then there is a partial loss of control effectiveness in u^{j_i}. In this thesis, partial actuator fault is considered, then the state equation of the driving motors in (4.6) with partial actuator faults for each robot can be written as follows (Zhang and Jiang, 2002):

$$x_{k+1} = A_k x_k + B_k u_k + \begin{bmatrix} b_1\gamma_k^1 & b_2\gamma_k^2 & \dots & b_m\gamma_k^{m_d} \end{bmatrix} \begin{bmatrix} u_k^1 \\ u_k^2 \\ \vdots \\ u_k^{m_d} \end{bmatrix} + w_k^x \tag{4.10}$$

or in compact form as,

$$x_{k+1} = A_k x_k + B_k u_k - B_k U_k \gamma_k + w_k^x \tag{4.11}$$

where:

$$U_k = \begin{bmatrix} u_k^1 & 0 & \dots & 0 \\ 0 & u_k^2 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & u_k^{m_d} \end{bmatrix}, \qquad \gamma_k = \begin{bmatrix} \gamma_k^1 \\ \gamma_k^2 \\ \vdots \\ \gamma_k^{m_d} \end{bmatrix}$$

In the absence of the knowledge of the loss of effectiveness factor γ_k, it can be modeled as a random bias vector:

$$\gamma_{k+1} = \gamma_k + w_k^x. \tag{4.12}$$

The bias augmented discrete linear state-space model then has the following form:

$$x_{k+1} = A_k x_k + B_k u_k + E_k \gamma_k + w_k^x \tag{4.13}$$

$$\gamma_{k+1} = \gamma_k + w_k^x \tag{4.14}$$

$$y_{k+1} = C_k x_{k+1} + \upsilon_{k+1} \tag{4.15}$$

The minimum variance solution to estimate the true values of the biases (the fault parameters) and states are obtained by applying the two-stage Kalman filter as follows (Zhang and Jiang, 2002):

The fault parameter estimator:

$$\hat{\gamma}_{k+1|k} = \hat{\gamma}_{k|k} \tag{4.16}$$

$$P^{\gamma}_{k+1|k} = P^{\gamma}_{k|k} + Q^{\gamma}_{k} \tag{4.17}$$

$$\hat{\gamma}_{k+1|k+1} = \hat{\gamma}_{k+1|k} + K^{\gamma}_{k+1}(\tilde{r}_{k+1} - H_{k+1|k}\hat{\gamma}_{k|k}) \tag{4.18}$$

$$K^{\gamma}_{k+1} = P^{\gamma}_{k+1|k}H^{T}_{k+1|k}(H_{k+1|k}P^{\gamma}_{k+1|k}H^{T}_{k+1|k}\tilde{S}_{k+1})^{-1} \tag{4.19}$$

$$P^{\gamma}_{k+1|k+1} = (I - K^{\gamma}_{k+1}H_{k+1|k})P^{\gamma}_{k+1|k} \tag{4.20}$$

The fault-free state estimator:

$$\tilde{x}_{k+1|k} = A_k\tilde{x}_{k|k} + B_k u_k + W_k\hat{\gamma}_{k|k} - V_{k+1|k}\hat{\gamma}_{k|k} \tag{4.21}$$

$$\tilde{P}^{x}_{k+1|k} = A_k\tilde{P}^{x}_{k|k}A^{T}_{k} + Q^{x}_{k} + W_k P^{\gamma}_{k|k}W^{T}_{k} - V_{k+1|k}P^{\gamma}_{k+1|k}V^{T}_{k+1|k} \tag{4.22}$$

$$\tilde{x}_{k+1|k+1} = \tilde{x}_{k+1|k} + \tilde{K}^{x}_{k+1}(y_{k+1} - C_{k+1}\tilde{x}_{k+1|k}) \tag{4.23}$$

$$\tilde{K}^{x}_{k+1} = \tilde{P}_{k+1|k} + C^{T}_{k+1}(C_{k+1}\tilde{P}^{x}_{k+1|k}C^{T}_{k+1} + R_{k+1})^{-1} \tag{4.24}$$

$$\tilde{P}^{x}_{k+1|k+1} = (I - \tilde{K}^{x}_{k+1}C_{k+1})\tilde{P}^{x}_{k+1|k} \tag{4.25}$$

The filter residual and its covariance:

$$\tilde{r}_{k+1} = y_{k+1} - C_{k+1}\tilde{x}_{k+1|k} \tag{4.26}$$

$$\tilde{S}_{k+1} = C_{k+1}\tilde{P}^{x}_{k+1|k}C^{T}_{k+1} + R_{k+1} \tag{4.27}$$

The coupling equations:

$$W_k = A_k V_{k|k} - B_k U_k \tag{4.28}$$

$$V_{k+1|k} = W_k P^{\gamma}_{k|k}(P^{\gamma}_{k+1|k})^{-1} \tag{4.29}$$

$$H_{k+1|k} = C_{k+1}V_{k+1|k} \tag{4.30}$$

$$V_{k+1|k+1} = V_{k+1|k} - \tilde{K}^{x}_{k+1}H_{k+1|k} \tag{4.31}$$

The compensated error and covariance estimator:

$$\hat{x}_{k+1|k+1} = \tilde{x}_{k+1|k+1} + V_{k+1|k+1}\hat{\gamma}_{k+1|k+1} \tag{4.32}$$

$$P_{k+1|k+1} = \tilde{P}^{x}_{k+1|k+1} + V_{k+1|k+1}P^{\gamma}_{k+1|k+1}V^{T}_{k+1|k+1} \tag{4.33}$$

The previous equations can be divided into two parts: the *time-update* equations and the *measurement-update* equations. The time-update equations, distinguished by the subscription $(k + 1|k)$, are responsible to obtain *a priori* estimate by moving the state and error covariances one step ahead in the time domain. The measurement-update equations, distinguished by the subscription $(k+1|k+1)$, are responsible to obtain *a priori* estimates through feedbacking measurements into the *a priori* estimates. The time-update equations are used for prediction, while measurement-update equations are used for correction. The whole prediction-correction process is used to estimate the states as close as possible to their real values. Figure 4.3 shows a schematic flow diagram of the two-stage Kalman filter.

4.2 Decision Making Algorithm

To facilitate the re-assignment, the proposed algorithms are implemented in a decentralized manner. The following assumptions are made for the FTCC algorithm:

Assumption 4.2. *There is no loss of communications between robots. Each robot in the team receives the position of other team members, i.e. each robot knows the position of other robots in the formation.*

The basic idea of the FTCC algorithm is to deal with the actuator faults in one or more robots according to the fault magnitude information γ^{j_i},

$$0 \leq \gamma^{j_i} \leq 1 \qquad i = 1, \ldots, m_d, \quad j = 1, \ldots, N \tag{4.34}$$

estimated by the above TSKF based FDD scheme to the faulty robot j.

Depending on the fault severity levels, the following situations may take place:

- If $\gamma^{j_i} = 0$, then all robots are fault-free. So all robots continue the planned mission.

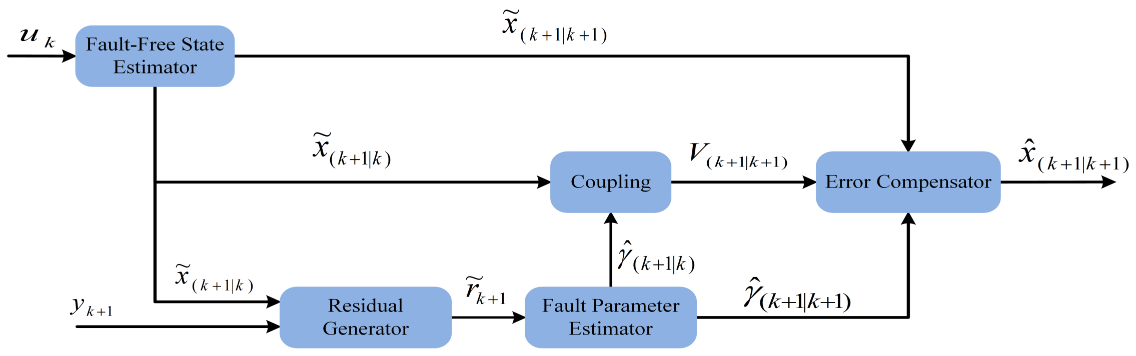


Figure 4.3: TSKF schematic diagram

- If $\bar{\gamma} \leq \gamma^{j_i} \leq 1$, then one or more robots are subject to severe actuator faults, and they are unable to complete the mission. So the remaining healthy robots start reconfiguring their formation. i.e, each robot switches to a new desired formation.
- If $0 < \gamma^{j_i} < \bar{\gamma}$, the mission can still be completed with degraded performance in the event of actuator faults in the robot team.

where $\bar{\gamma}$ is the critical value of the loss of effectiveness, the fault can be considered as a severe fault if the loss of effectiveness is equal to $\bar{\gamma}$ or higher. It is worth mentioning that $\bar{\gamma}$ is varied depending on the type of robot, and the type of mission.

4.2.1 Task Re-Assignment Algorithm Based on the Graph Theory

The formation reconfiguration is investigated based on Graph Theory (Desai et al., 1999). Considering that the internal behavior of the team is described by the pair $(r, \mathcal{H})$, where r describes the formation shape, and $\mathcal{H}$ is the control graph representing the control strategy used by each robot. Graphs are made of edges and vertices. Each vertex in the graph corresponds to a robot, and the edges describe the dependencies of each robot on the adjacent robots.

According to this theory, the control graph $\mathcal{H}$ is represented as an $N \times N$ adjacency matrix G, where N is the number of robots in the formation. The elements of matrix G are either 0 or 1. 1 in the (i, j) entry represents an direct edge from robot i to robot j, and 0 represents no edge between the robot i and j which means that the motion of robot j is independent on robot i.

Remark 4.2. *Every directed edge in the graph goes from a lower vertex label to a higher vertex label.*

The appearance of a 1 in a column for a robot defines its controller as (Desai et al., 1999):

$$\sum_{columns} 1's = \begin{cases} 0 & \text{leader} \\ 1 & \text{follower with } l-\psi \text{ control} \\ 2 & \text{follower with } l-l \text{ control} \end{cases}$$

where $l-\psi$ control means that one robot follows another by controlling the relative distances between them (as a leader-follower case), and $l-l$ control means that the robot maintaining a specified distance from two robots.

To apply the Graph Theory, the initial and final formation configurations are represented by the adjacency matrices G and H respectively. The transition from one control graph to another is presented by a transition matrix T, where $T = H - G$. There are 3 possible values of the (i, j) elements in the matrix T

$$\begin{cases} 0 & \text{no edge connection between } i \text{ and } j \\ -1 & \text{the edge connection needs to be broken} \\ 1 & \text{new edge needs to be established.} \end{cases}$$

It is noted that the matrices G and H have the same dimensions, but the matrix H represent the formation configuration of the healthy robots. So, it is assumed that during formation reconfiguration the faulty robots still exist but there is no edge connection between them and the other robots, *i.e.* the element of the columns represent the faulty robots will be zeros. So, the matrix T can be calculated and the remaining robots start reconfiguring their formation shape. Figure 4.4 presents the idea of FTCC based on the Graph Theory.

Remark 4.3. *The Graph Theory focuses on the problem of achieving the desired formation, not how to generate the reconfigured formation. So, it is assumed that the final formation shape is known and each robot already knows its location in all faulty cases. i.e. all the*

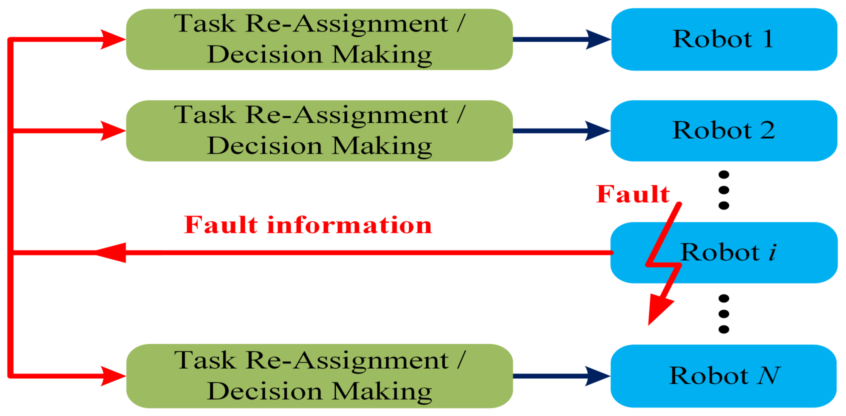

Task Re-Assignment / Decision Making
Robot 1
Task Re-Assignment / Decision Making
Robot 2
Fault
Fault information
Robot i
Task Re-Assignment / Decision Making
Robot N

Figure 4.4: Task re-assignment mechanism based on the Graph Theory

possible formation reconfigurations are already embedded in each robot controller. The main advantage of this algorithm is the decentralization, where each robot can take its own decision according to the received fault information as shown in Figure 4.4. However, with increasing the number of robots, the system will be complicated to apply in real-time. Accordingly, another method is proposed to avoid this drawback as will be explained later.

4.2.2 Task Re-Assignment Based on the Optimal Assignment

In case of severe fault occurrence, the FTCC problem is solved as an assignment problem. The basic idea of optimal assignment is to ensure that during formation reconfiguration one and only one robot should be assigned to a unique place in the new formation shape. Figure 4.5 explains the proposed algorithm.

Once the leader receives the fault decision from the FDD unit, the leader knows that the remaining number of followers is $F = N - 1$. The leader sends a new formation shape S parametrized by a vector r in Cartesian coordinates relative to the leader position. The desired formation of F followers is assumed as slots to be filled, whilst each follower needs to be assigned to only one of the slots. This can be formulated as an optimal assignment problem, where the cost function $c_{ij} = c(F_i, S_j)$ decides the cost of assigning the robot F_i to slot S_j. The cost considered herein is to minimize the distance between the follower and the assigned slot. Also the leader has a pre-defined formation shape according to the number of remaining robots. Figure 4.6 shows the possible formation shapes for six robots or fewer.

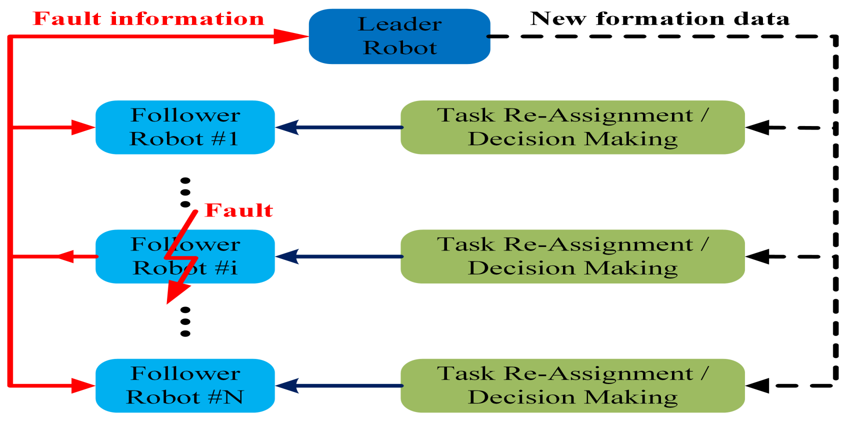

Fault information
Leader Robot
New formation data
Follower Robot #1
Task Re-Assignment / Decision Making
Fault
Follower Robot #i
Task Re-Assignment / Decision Making
Follower Robot #N
Task Re-Assignment / Decision Making

Figure 4.5: Task re-assignment algorithm based on the optimal assignment

The optimal assignment problem can be mathematically formulated as follows:

Definition 4.1. *Let $F = \{F_1, F_2, \ldots, F_{n_R}\}$ denote the healthy followers and $S = \{S_1, S_2, \ldots, S_{n_R}\}$ denote the slots. Given an $n_R \times n_R$ cost matrix where the element at the i^{th} row and the j^{th} column corresponds to the cost of assigning the i^{th} follower to the j^{th} slot, find a permutation π of $\{1, 2, \ldots, n_R\}$ for which*

$$\sum_{i=1}^{n_R} c(F_i S_{\pi(i)}) \tag{4.35}$$

is a minimum.

Let $x_{ij} = 1$ denote F_i occupying S_j and 0 otherwise. Then the optimization objective function is

$$\min \sum_{i=1}^{n_R} \sum_{j=1}^{n_R} c_{ij} x_{ij}, \tag{4.36}$$

subject to

$$\sum_{i=1}^{n_R} x_{ij} = 1 \quad \forall j, \qquad \sum_{j=1}^{n_R} x_{ij} = 1 \quad \forall i \tag{4.37}$$

The above constraints presented in Eq. (4.37) ensure unique assignment, i.e., one robot occupies one and only one slot. Many algorithms are presented in operations research and network theory. The most common algorithm is the Hungarian algorithm that reduces the complexity of finding the optimal assignment from combinatorial to polynomial in time (Papadimitriou and Steiglitz, 1998). The input to this algorithm is the $n_R \times n_R$ cost matrix

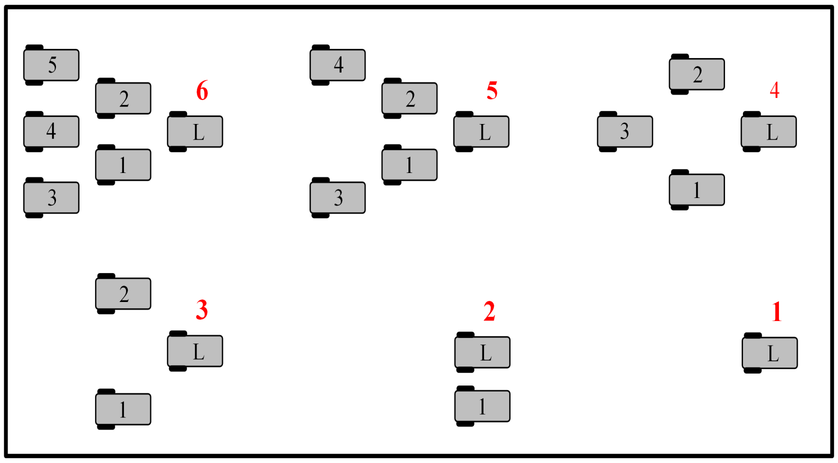


Figure 4.6: Possible formation shapes for six or fewer WMRs

which is defined in Definition 4.1. The solution can be obtained using a bipartite graph where there are n_R vertices representing the healthy followers, n_R vertices representing the slots, and edges connect the followers and slots where each edge has a non-negative cost c_{ij}. The Hungarian algorithm obtains the set of edges that minimize the sum of edge costs such that there is an edge connecting each follower to a unique slot.

4.2.3 Motion Re-Coordination

In case of faulty robot is still able to complete the mission, the other healthy robots should reconfigure their controllers within the capability of the faulty robot. The problem that the leader tracks a pre-defined trajectory. So, the pre-defined trajectory should be updated considering this faulty situation.

The idea of motion re-coordination is that: once the leader receives the faulty signal from the faulty robot and it is able to complete the mission regardless the fault, then the leader will re-generate its desired trajectory corresponding to the fault severity γ^{j_i}, i.e. it re-generates the values of x_r, y_r, $\dot{x}_r$, and $\dot{y}_r$ as follows:

$$\tilde{z}_r = \left[\tilde{x}_r \; \tilde{y}_r \; \tilde{\dot{x}}_r \; \tilde{\dot{y}}_r\right]^T = \left(1 - \gamma^{j_i}\right) \left[x_r \; y_r \; \dot{x}_r \; \dot{y}_r\right]^T \tag{4.38}$$

where $\tilde{x}_r$, $\tilde{y}_r$, $\tilde{\dot{x}}_r$, and $\tilde{\dot{y}}_r$ are the states of the re-planned trajectory.

The leader controller will get the leader to converge to the new trajectory. That means that the leader controller should be reconfigured to meet the following objective:

$$\lim_{t \to \infty} \left(\tilde{z}_r(t) - z_l(t)\right) = 0, \tag{4.39}$$

Following the steps presented in Section 3.3, the followers' controllers will ensure the followers to follow the leader as required in Eq. (3.40) considering the new capabilities of the leader and the faulty robot. Consequently, the performance of the whole team will be degraded due to the fault. Thus, the mission can still continue but with degraded performance. Such a control strategy is referred as to graceful performance degradation as proposed in (Zhang and Jiang, 2003) for single vehicle cases. The FTCC strategies for a team of WMRs presented in Sections 4.2.2 and 4.2.3 are summarized in Algorithm 4.1.

Algorithm 4.1 FTCC strategy for a team of WMRs

```
1: for each follower do
2:     detect $\gamma^{j_i}$
3:     send $\gamma^{j_i}$ to the leader
4:     if $0 < \gamma^{j_i} < \bar{\gamma}$ then
5:         for the leader do
6:             receive $\gamma^{j_i}$ from the faulty robot
7:             regenerate the new reference trajectory states $\tilde{z}_r$
8:             follow the new reference trajectory
9:         end for
10:        for each healthy follower do
11:            update the reference trajectory states according to the new capabilities of the leader and the desired formation.
12:        end for
13:    end if
14:    if $\bar{\gamma} \leq \gamma^{j_i} \leq 1$ then
15:        for the leader do
16:            determine the remaining number of the healthy robots $F = N - 1$
17:            send the new formation data $S$ to the followers
18:        end for
19:        for each healthy follower do
20:            calculate the cost matrix $c$
21:            apply the Hungarian algorithm
22:            assign to the corresponding slot
23:        end for
24:    end if
25: end for
```

4.3 Numerical Validation via Simulation

To verify the proposed control strategies presented in this Chapter, a team WMRs are used in simulation to validate the effectiveness of the proposed control algorithm. All the values adopted for the FTCC control system parameters are shown in Table 4.1.

Table 4.1: FTCC control system parameters

Parameter	Value	Parameter	Value
I_m	1.07 gm.cm^2	b	3.5077*10^{-6} N.m.s
k_b	0.0235 V/rad/sec	k_t	0.0235 N.m/Amp
R_a	2.06 ohm	L_a	0.238 mH

Remark 4.4. *All the values of the robots' control system parameters as well as the collision avoidance algorithm parameters are the same as those mentioned in Table 3.1.*

The covariance matrices for the TSKF are set to:

$$Q^x = \begin{bmatrix} 0.01 \times I_{2\times2} & 0_{2\times2} \\ 0_{2\times2} & 0.01 \times I_{2\times2} \end{bmatrix},$$
$$R^x = 0.01 \times I_{2\times2},$$
$$Q^\gamma = 0.001 \times I_{2\times2}$$

Different cases have been carried out to prove the robustness of the proposed algorithms. These cases are:

- Case 1: A severe fault occurs in the leader robot, in which the Graph Theory is applied;
- Case 2: A severe fault occurs in the second follower, while the optimal assignment is applied; and
- Case 3: A fault occurs in the second follower, however it can still complete the mission.

Remark 4.5. *As mentioned in Remark 4.3,* FTCC *based on the Graph theory is difficult to apply in real-time experiment. Therefore, Case 1 will be investigated only in simulation, On the other hand, Cases 2 and 3 will be validated later in real-time experiments. As a result,* FDD *algorithm will be investigated in simulation and real-experiments only in Cases 2 and 3. While, in case 1, only the task re-assignment will be assessed, assuming that the fault is already detected and diagnosed.*

Remark 4.6. *In this thesis, it is assumed that critical value of the loss of effectiveness $\bar{\gamma}$ is 0.65.*

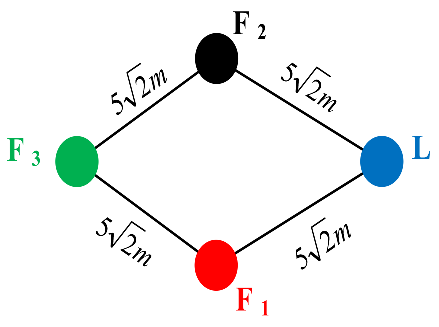
4.3.1 Case 1: A Severe Fault Occurs in the Leader Robot, While the Graph Theory is Applied

Simulation Scenario

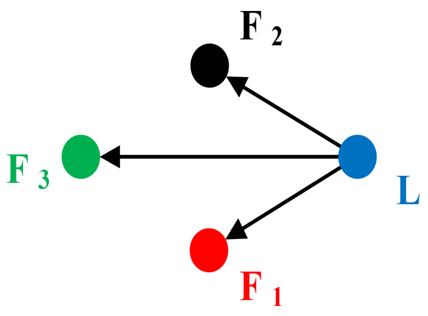
In this case, a team of four WMRs performs a diamond formation as shown in Figure 4.7(a). The leader's initial position is $q_l(0) = [1, -1, \pi/3]^T$. The first follower's initial position is $q_1(0) = [-5, -7, \pi/3]^T$, and its desired formation with respect to the leader is F_1^d is $5\sqrt{2}$ m and 225°. The second follower's initial position is $q_2(0) = [-5, 3, \pi/3]^T$ and its desired formation with respect to the leader is F_2^d is $5\sqrt{2}$ m and 135°. The third follower's initial position is $q_3(0) = [-9, -2, \pi/3]^T$ and its desired formation with respect to the leader is F_3^d is 5 m and 180°. The initial control graph of the team is shown in Figure 4.7(b). Based on this initial control graph, the matrix G will be:

$$G = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

It is assumed that the left actuator of the leader suffers a severe fault ($\gamma^{l_L} = 0.9$ (90%) at time instant $t = 80$ sec, which leads to the mission incompletion.



(a) Initial formation configuration



(b) Initial control graph

Figure 4.7: Simulation Scenario of Case 1

Task Re-Assignment Results

Assuming that the fault is detected and diagnosed, the leader sends the fault data to the rest of robots. Then, it will be separated from the formation. Under this fault situation, the formation reconfigured to a triangular formation.

Based on the embedded data in each robot controller, the first follower will replace the leader, and the third follower will replace the first one as shown in Figure 4.8(b). As a result, the matrix H will be:

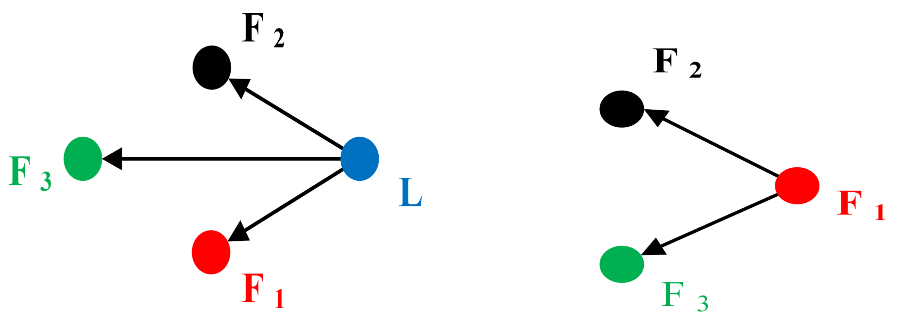
$$H = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

and the transition matrix T is:

$$T = \begin{bmatrix} 0 & -1 & -1 & -1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

meaning that the first follower will establish new edge connections with the second and third followers, while the leader robot will break all the edge connections with the other robots.

Figures 4.9 and 4.10 illustrate the robots' trajectories during mission execution. As shown in these figures, the first follower will replace the leader, and the third follower will



(a) Initial control graph (b) Final control graph

Figure 4.8: Initial and final control graphs of Case 1

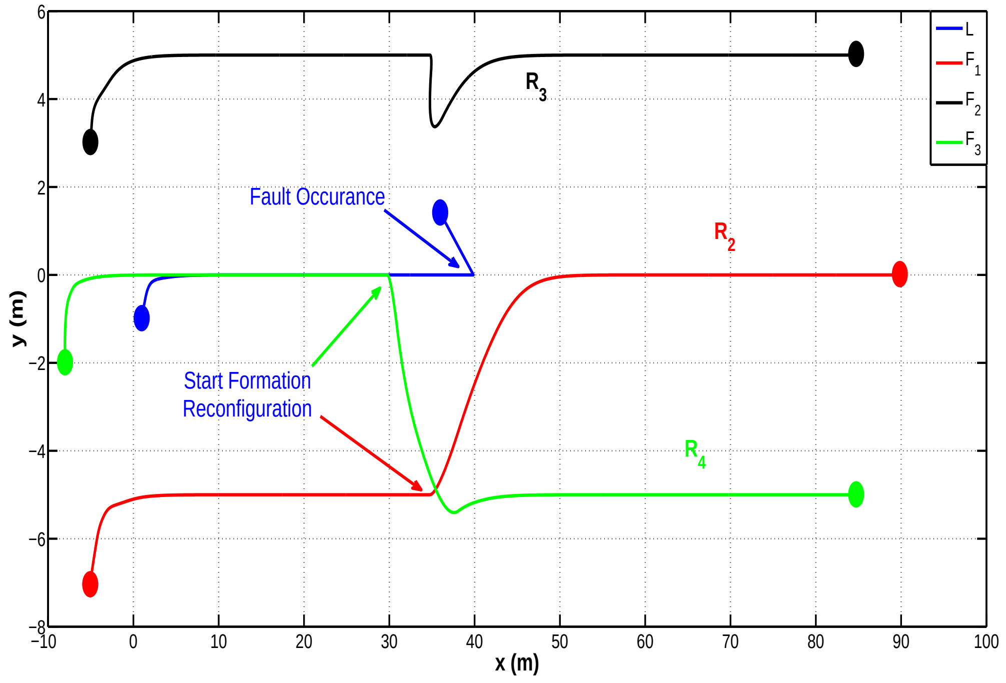


Figure 4.9: Robots' trajectories during mission execution in Case 1

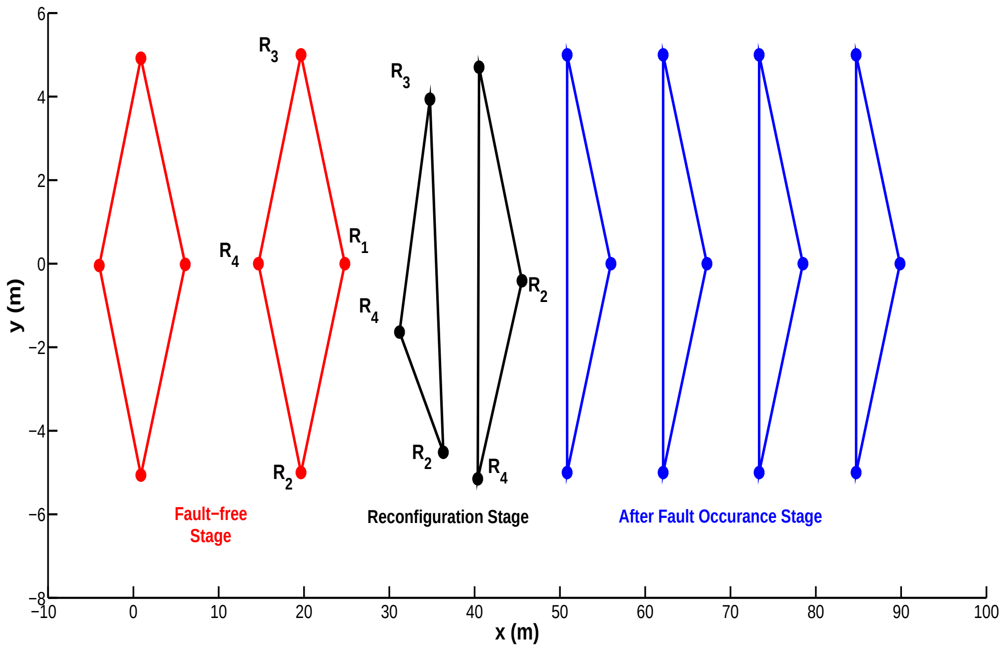


Figure 4.10: Snapshot of the formation of the team in Case 1

replace the first one. Figure 4.9 shows the effectiveness of formation controller in both fault-free and faulty cases. It is clear that the robots converge to their desired trajectories. Figure 4.10 presents snapshots of the team formation configuration during mission execution which indicate fault-free, reconfiguration, and after fault occurrence stages.

4.3.2 Case 2: A Severe Fault Occurs in the Second Follower, While the Optimal Assignment is Applied

Simulation Scenario

In this case, the following scenario is considered. A team of five WMRs performs a triangular formation as shown in Figure 4.11. The leader's initial position is $q_l(0) = [1, -1, \pi/3]^T$. The first follower's initial position is $q_1(0) = [-2, -3, \pi/3]^T$, and its desired formation with respect to the leader F_1^d is $2\sqrt{2}$ m and 225°. The second follower's initial position is $q_2(0) = [-2, 2, \pi/3]^T$ and its desired formation with respect to the leader F_2^d is $2\sqrt{2}$ m and 135°. The third follower's initial position is $q_3(0) = [-4, -5, \pi/3]^T$ and its desired formation with respect to the leader F_3^d is $4\sqrt{2}$ m and 225°. The fourth follower's initial position is $x_4(0) = [-4, 4, \pi/3]^T$ and its desired formation with respect to the leader F_4^d is $4\sqrt{2}$ m and 135°. It is assumed that the right actuator of the second follower suffers a severe fault ($\gamma^{2_R} = 0.85$ (85%) as shown in Figure 4.12) at time instant $t = 45$ sec, which leads to the mission incompletion.

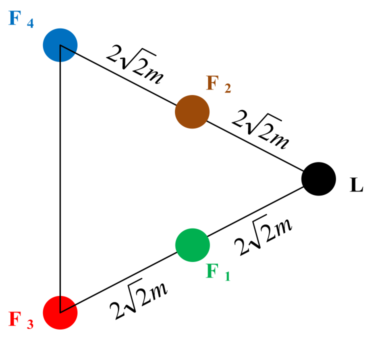


Figure 4.11: The configuration of WMR formation during simulation of Cases 2 and 3

FDD Results

According to the faulty situation, the value of the loss of effectiveness of the actuators in the second follower are detected, and the fault magnitude is estimated by the proposed FDD scheme as showed in Figure 4.12. As can be viewed in Figure 4.12, the estimate of γ^{2_R} converges to the real value 0.85 after 2 seconds of the fault occurrence with a steady-state error of almost 0.0065. Also, the estimate of γ^{2_L} remains close to zero. As shown in Figure 4.13, at t = 45 sec, the right motor of the second follower is subjected to a severe fault, resulting in a loss of its effectiveness by about 85%. Due to this faulty situation, and according to the proposed FTCC algorithm, the robot starts getting out from the formation, and stops accordingly.

Task Re-assignment Results

Once the fault is detected and diagnosed, the second follower is separated from the formation sending a signal to the leader. Under this fault situation, the leader sends new formation data to the remaining three followers as desired slots' coordinates with respect to the leader position. These data are:

$$S_1 = [-2, -2], \quad S_2 = [-2, 2], \quad S_3 = [-4, 0]$$

Subsequently, the embedded Hungarian algorithm in each follower builds the cost matrix which is the distance between each follower and each slot. The updated cost matrix is:

$$c = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{31} & c_{32} & c_{33} \\ c_{41} & c_{42} & c_{43} \end{bmatrix} = \begin{bmatrix} 0 & 4 & 2\sqrt{2} \\ 2\sqrt{2} & 2\sqrt{10} & 4 \\ 2\sqrt{10} & 2\sqrt{2} & 4 \end{bmatrix}$$

According to the Hungarian algorithm, the optimum assignment is:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

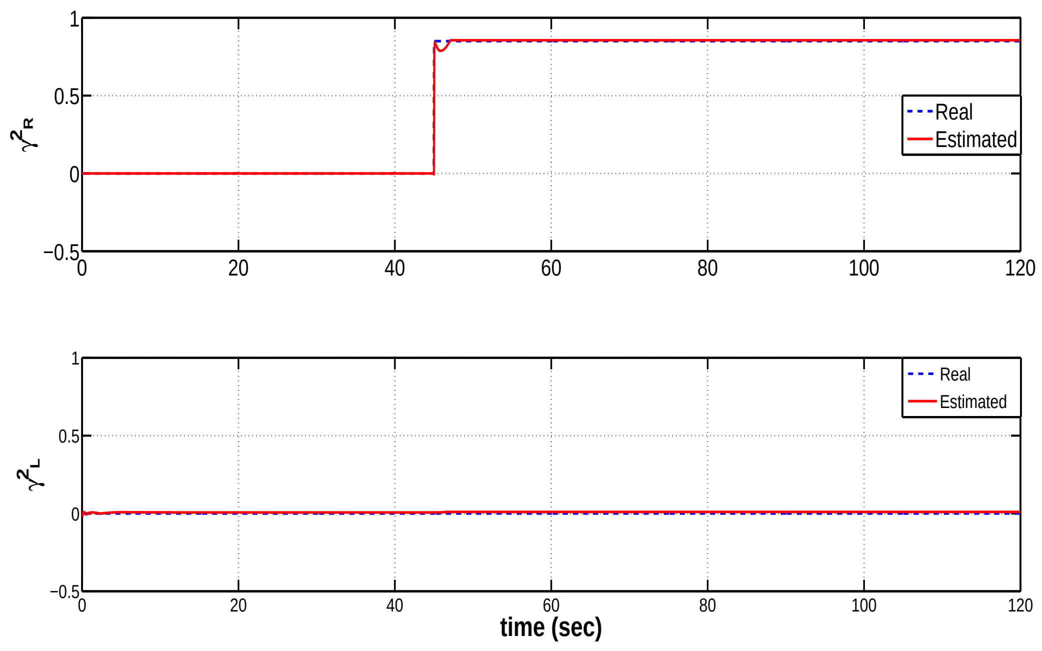


Figure 4.12: The true and estimated values of the loss of effectiveness of the second follower's actuators in Case 1

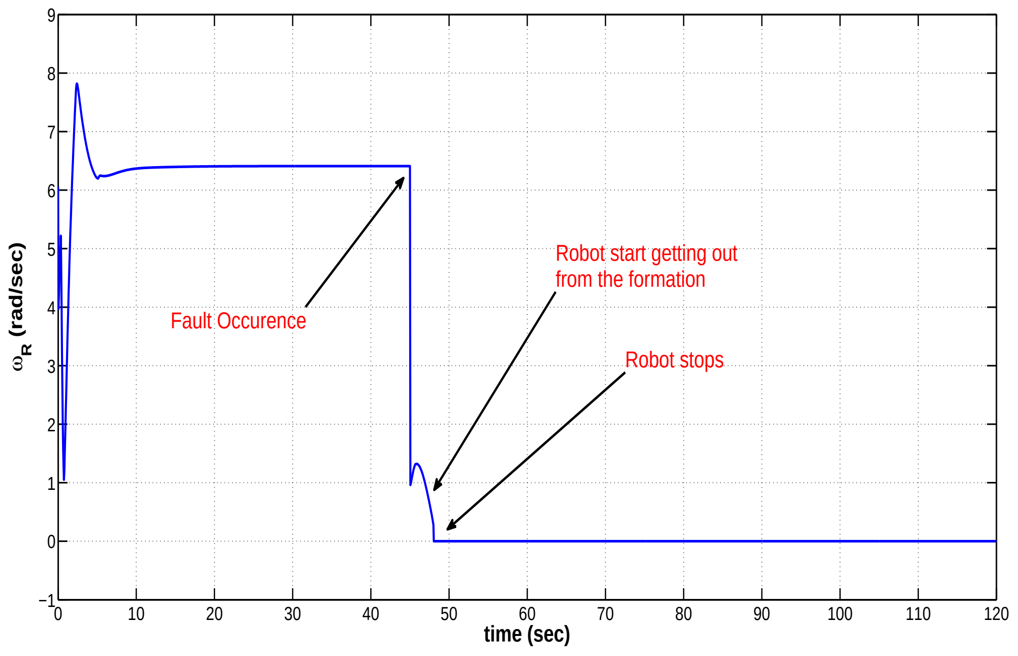


Figure 4.13: The output of the right actuator in Case 1

which means that the first follower is assigned to slot S_1 (i.e the first follower will not change its location), the third follower is assigned to the slot S_3, and the fourth robot to the slot S_2, as shown in Figure 4.14.

Figures 4.15 and 4.16 illustrate the robots' trajectories during mission execution. According to the fault situation, and based on the results obtained from the Hungarian algorithm, the third and the fourth followers start changing their positions with respect to the leader to change the whole formation to be a diamond shape. Figure 4.15 shows the effectiveness of formation controller in both fault-free and faulty cases. It is clear that the robots converge to the desired trajectories in the fault-free case and the healthy robots change their positions and follow a new formation pattern. Figure 4.16 presents snapshots of the team formation configuration during mission execution which indicate fault-free, reconfiguration, and after fault occurrence stages.

4.3.3 Case 3: A Fault Occurs in the Second Follower, However it Can Still Complete the Mission

Following the same initial configuration scenario of Case 2 (shown in Figure 4.11), it is assumed in this case that the right actuator of the second follower encounters a fault (γ^{2_R} = 0.35 (35%) as shown in Figure 4.17) at time instant t = 45 sec. However it is still able to continue the mission with degraded performance.

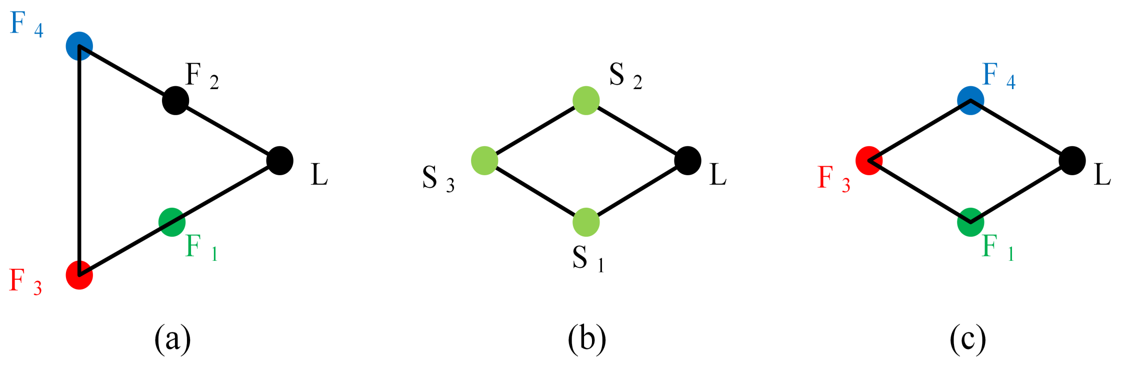


Figure 4.14: (a) Initial formation, (b) Desired formation, (c) Final formation

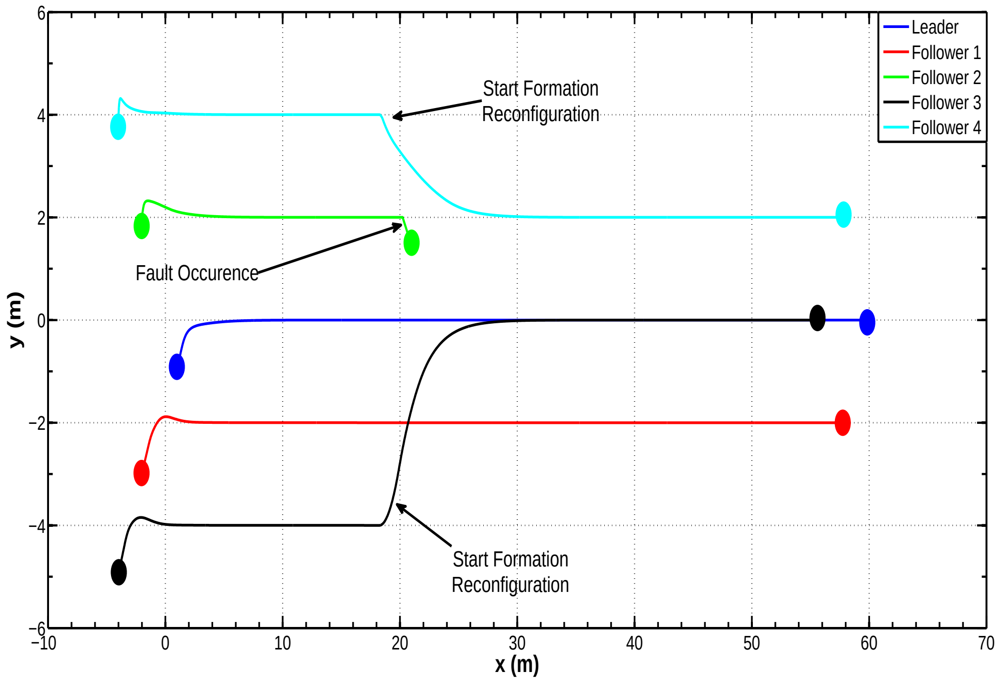


Figure 4.15: Robots' trajectories during mission execution in Case 2

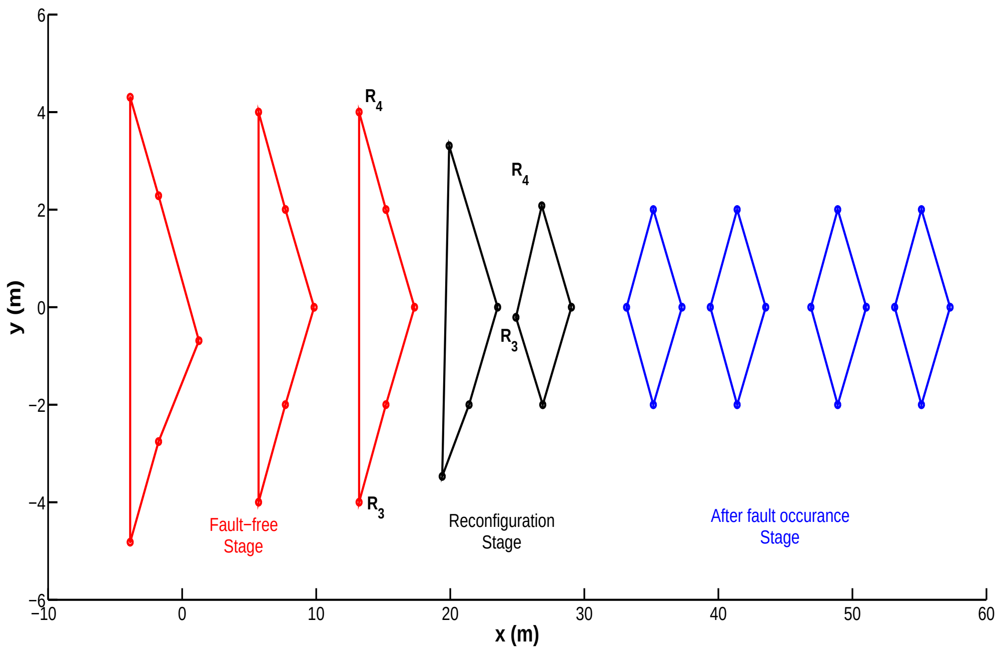


Figure 4.16: Snapshot of the formation of the team in Case 2

FDD Results

According to this situation, FDD scheme isolates the fault and estimate the value of the loss of effectiveness of the faulty actuator of the second follower as shown in Figure 4.17. As can be viewed in Figure 4.17, the estimate of the fault magnitude by means of the two-stage Kalman filter converges to the real value of 0.35 (35%) after 3 sec of the fault occurrence with a steady-state error of almost 0.007. Also, the estimate of γ^{2_L} remains close to zero. From Figure 4.18, at $t = 45$ sec, the right motor of the second follower is subjected to a fault, leading to a loss of its effectiveness by about 35%. The robot continues the motion with 65% of its capability.

Motion Re-Coordination Results

Once the fault is detected and diagnosed, the second follower sends the value of the loss of effectiveness γ^{2_R} to the leader. Consequently, the leader updates the reference trajectory states according to Eq. (4.38). As shown in Figure 4.19, once the second robot subject to a fault in its right motor, it tries to compensate this fault and continue the mission.

With comparison of Figure 4.15 and Figure 4.19, and knowing that the simulation time is the same for both cases, it can be noticed that in Case 2, the leader travels a distance less than Case 1. That is due to the fact that the leader (and the other team members) reduces their capabilities by considering the fault occurred at the second follower and its reduced capability. If not, the other four robots will continue the mission with their normal capabilities, but the second follower cannot keep the desired formation relative to the leader.

4.4 Experimental Results Analysis

Due to the space limitation in the laboratory, the experiment is performed with one leader and two followers only. In the experiments, the team of three WMRs performs a triangular formation.

The leader's initial position is $q_l(0) = [-0.19, -1.92, 100°]^T$. The initial positions of the followers are $q_1(0) = [-0.69, -2.45, 100°]^T$ and $q_2(0) = [-0.69, -1.474, 100°]^T$, respectively.

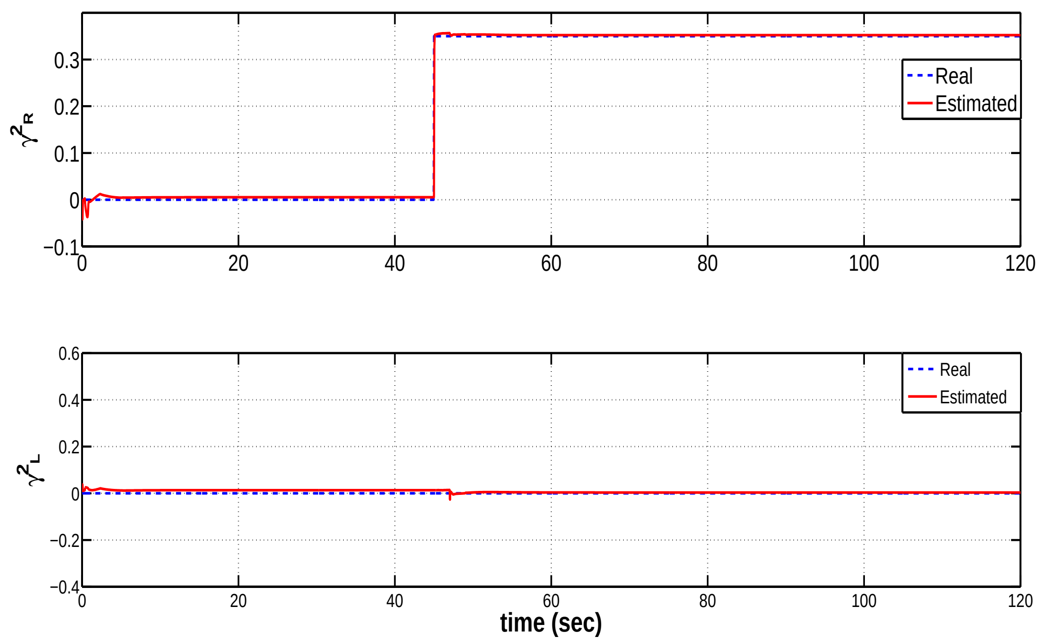


Figure 4.17: The true and estimated values of the loss of effectiveness of the second follower's actuators in Case 3

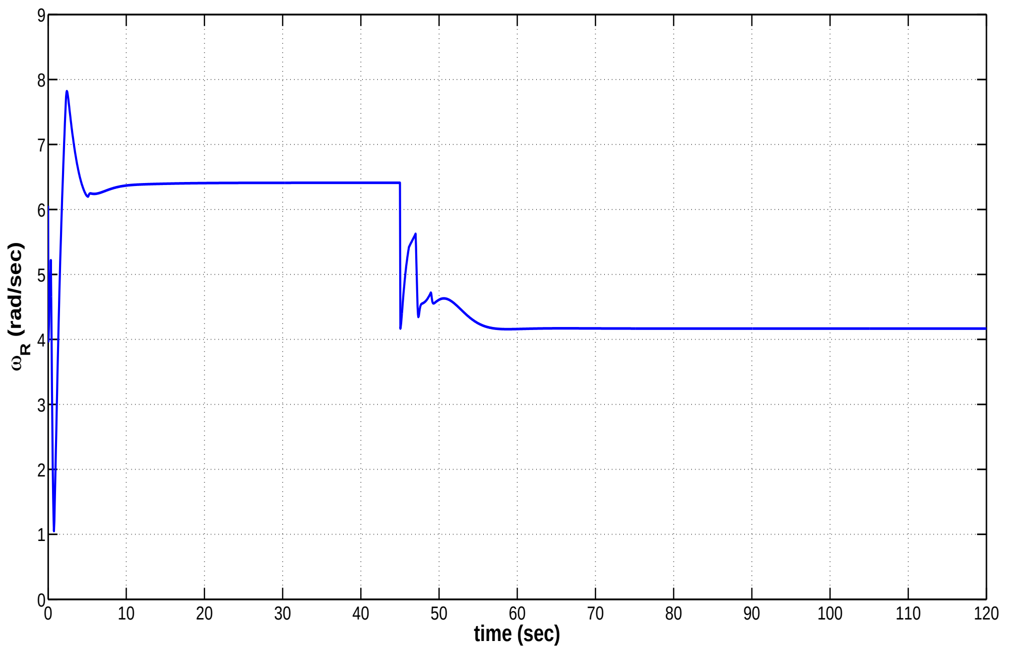


Figure 4.18: The output of the right actuator in Case 3

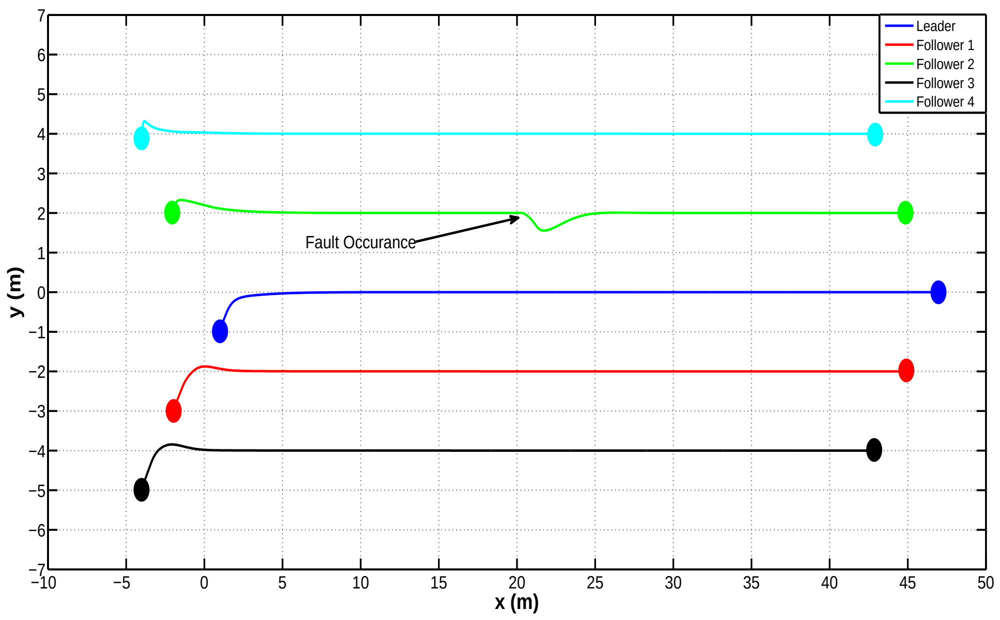


Figure 4.19: Robots' trajectories during mission execution considering the fault occurred in the second follower in Case 3

The desired formation with respect to the leader is 0.75 m and 225° for the first follower, and 0.75 m and 135° for the second one. As the same in Cases 2 and 3 in numerical simulation, severe fault and non-severe fault occurred in the second follower. The leader tracks a reference trajectory defined as:

$$x_r(t) = \cos(0.25t) - 0.1 \tag{4.40}$$

$$y_r(t) = \sin(0.25t) - 0.6 \tag{4.41}$$

4.4.1 Analysis of Experimental Results of Case 1

In this case, at time instant $t = 12$ sec, a severe fault is injected to the left motor of the second follower, leading to mission incompletion of this robot.

FDD Results

Under this faulty situation, FDD algorithm detects the fault occurred at the left motor of the second robot, and estimates the value of the loss of effectiveness γ^{2_L}. Figure 4.20

shows that the two-stage Kalman filter can estimate the real value of loss of effectiveness γ^{2_L} within 2 sec of fault occurrence. Also, the estimate of the fault magnitude of the loss of effectiveness of the right actuator γ^{2_R} remains close to zero. As can be seen from Figure 4.21, the left motor of the second follower is subject to a severe fault, leading to a loss of its effectiveness by about 92%. The robot cannot continue its planned mission and gets out from the formation.

Task Re-assignment Results

Once the fault is detected and diagnosed, the second follower stops and sends a signal to the leader. Under this fault situation, the leader sends new formation command to the remaining healthy robot. According to the possible formation patterns presented in Figure 4.6, new formation data are sent to the first follower which is 0.75 m and 270°. Figure 4.22 illustrates the robots' trajectories during mission execution. As can be observed, the team starts the formation in a triangular shape, and ending in the form of line formation due to the fault occurrence of the fault in the second follower robot at 12 sec. Moreover, Figure 4.23 illustrates the desired formation angle of the first follower. From Figure 4.23, the first follower starts reconfiguring its formation once it receives the new desired formation command.

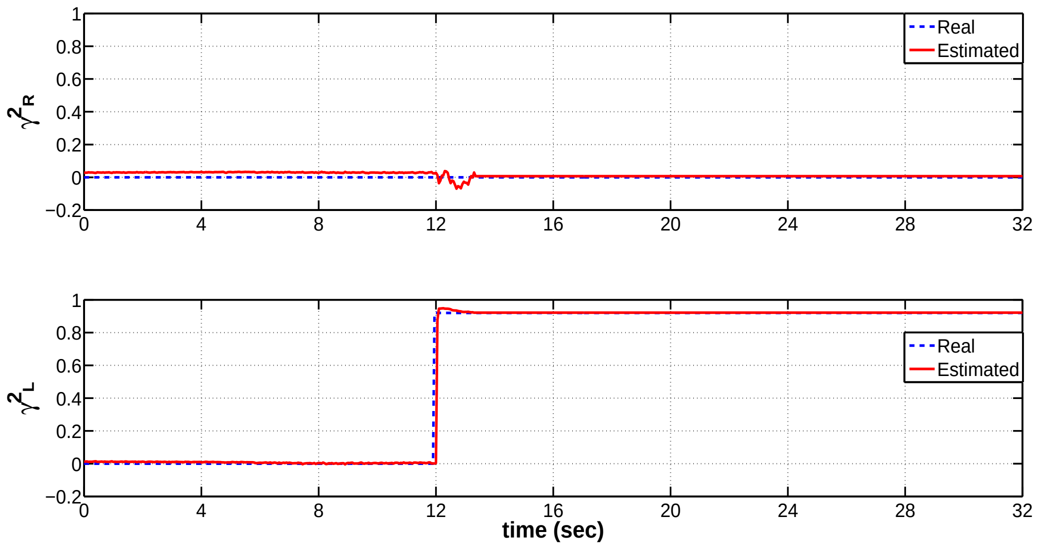


Figure 4.20: The true and estimated values of the loss of effectiveness γ^{2_L} in Case 1

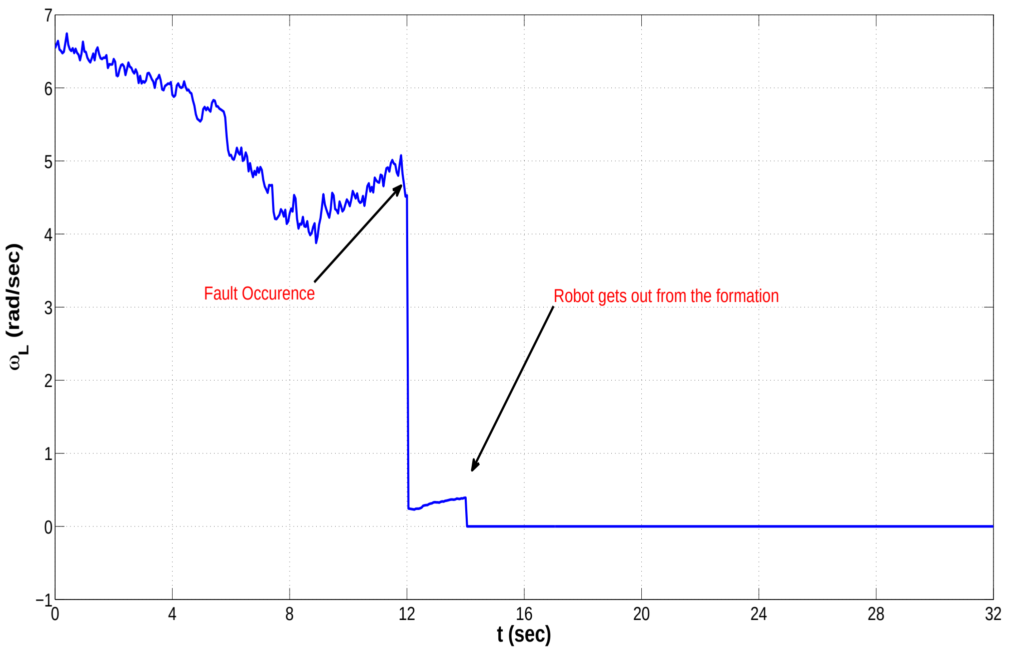


Figure 4.21: The output of the left motor in Case 1

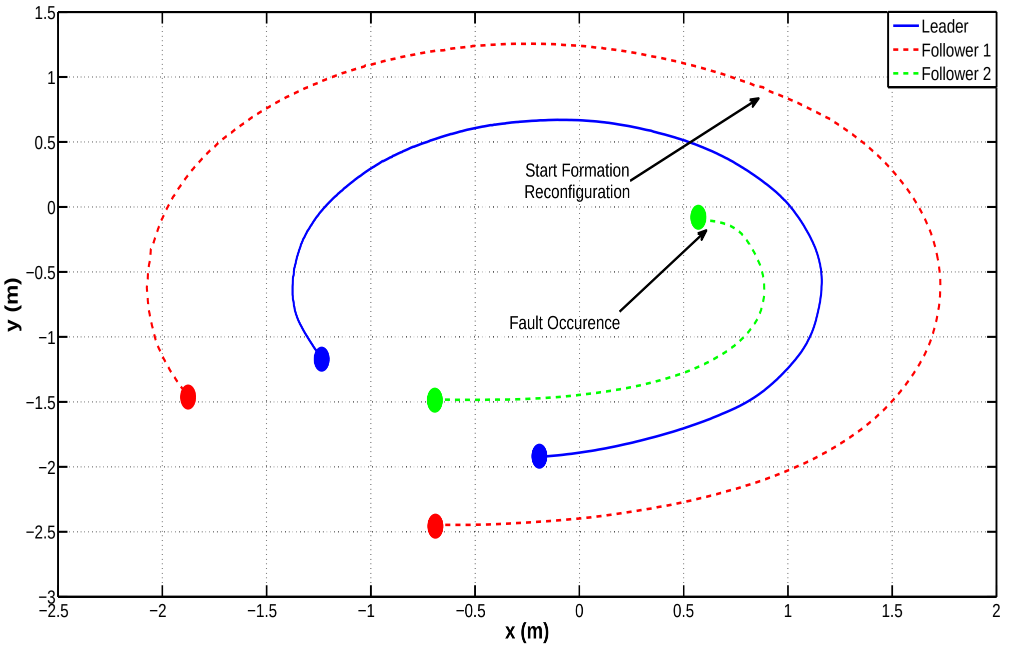


Figure 4.22: Robots' trajectories during mission execution considering the severe fault occurred in the second follower in Case 1

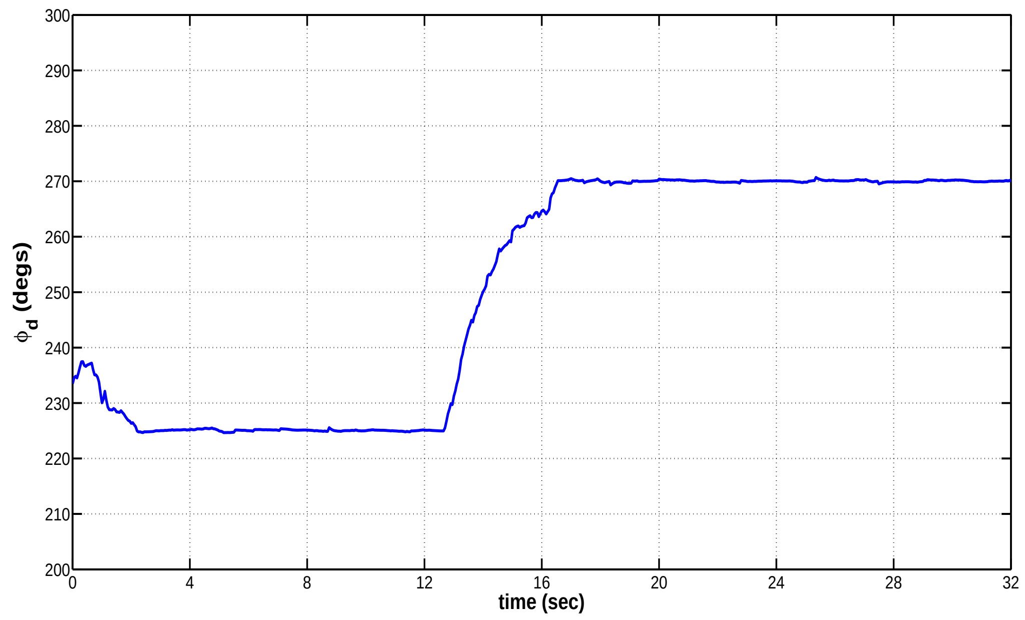
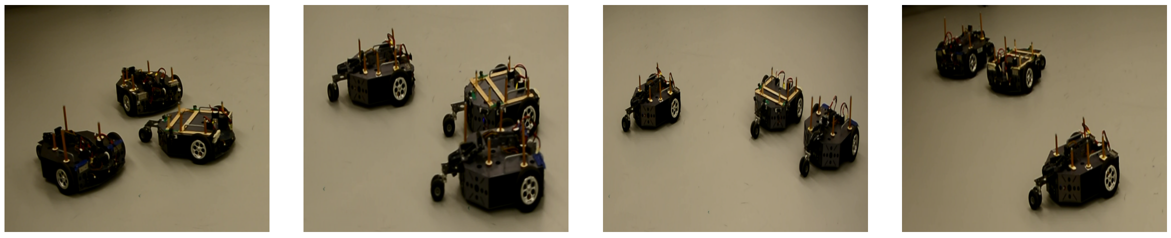


Figure 4.23: Desired formation angle of the first follower showing the formation reconfiguration according the faulty situation in Case 1

Figure 4.24 presents snapshots of the real experiment. When $t = 9$ sec, the robots still perform the desired formation configuration in the normal condition. When $t = 13$ sec, the first follower receives the new formation data, and starts reconfiguring its position relative to the leader according to the new situation. At $t = 16$ sec, the second follower completely gets out from the formation, and the remaining two followers continue the mission, while the first follower changes its position corresponding to the new formation configuration with a parallel motion pattern. When $t = 32$ sec, the leader and the first follower ends the planned mission.



(a) $t = 9$ sec (b) $t = 14$ sec (c) $t = 19$ sec (d) $t = 32$ sec

Figure 4.24: Snapshots of the experiment in Case 1

4.4.2 Analysis of Experimental Results of Case 2

In this scenario, at time instant $t = 13$ sec, a fault is injected to the left motor of the second follower. The loss of effectiveness of the left motor is about 38%. So, the second follower is still capable of continuing the mission but with degraded performance.

FDD Results

Under this situation, the fault is detected and the value of the loss of effectiveness γ^{2_L} is estimated by means of the FDD algorithm. It is illustrated in Figure 4.25 that the two-stage Kalman filter can estimate the real value of the loss of effectiveness γ^{2_L} within 2 sec of fault occurrence. Also, the estimate of γ^{2_R} remains close to zero. Based on Figure 4.26, the left motor of the second follower is subject to a fault, resulting in a loss of its effectiveness by about 38%. The robot continues its planned mission with 62% of its capability.

Motion Re-Coordination Results

Based on FDD result, the second follower sends the fault information to the leader. As a result, and according to Eq. (4.38), the leader updates the reference trajectory states. From Figure 4.27, the second follower tries to accommodate this fault and continue the mission, together with leader and the first follower.

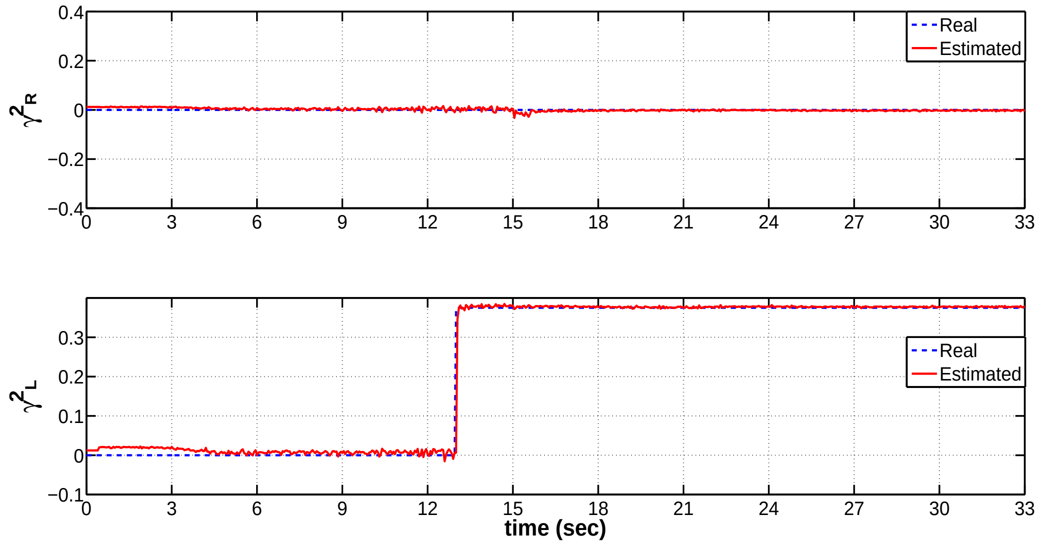


Figure 4.25: The true and estimated values of the loss of effectiveness γ^{2_L} in Case 2

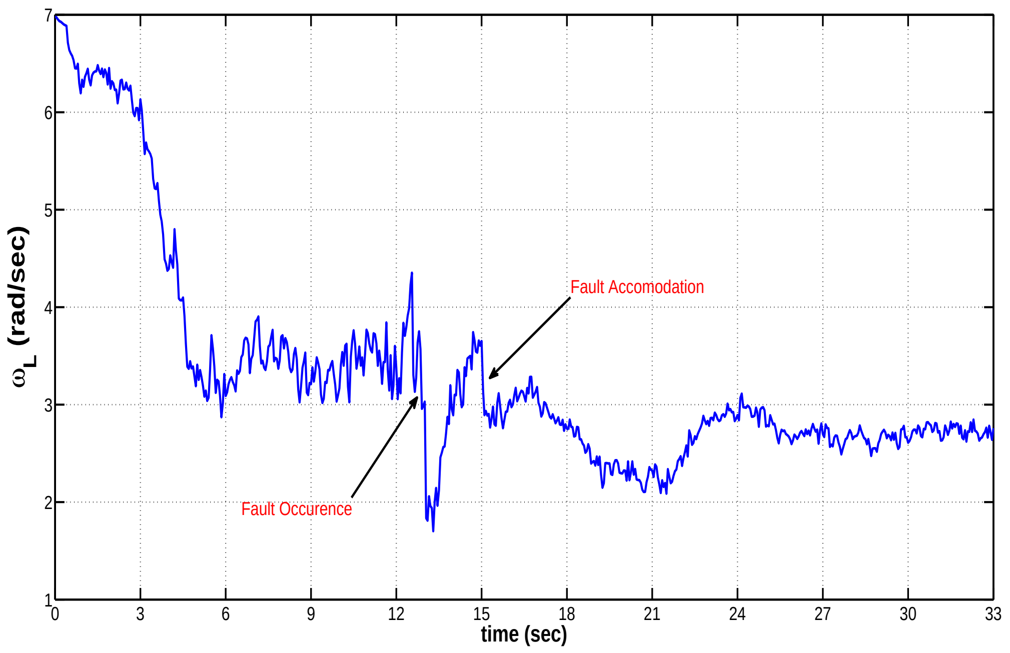


Figure 4.26: The output of the left motor in Case 2

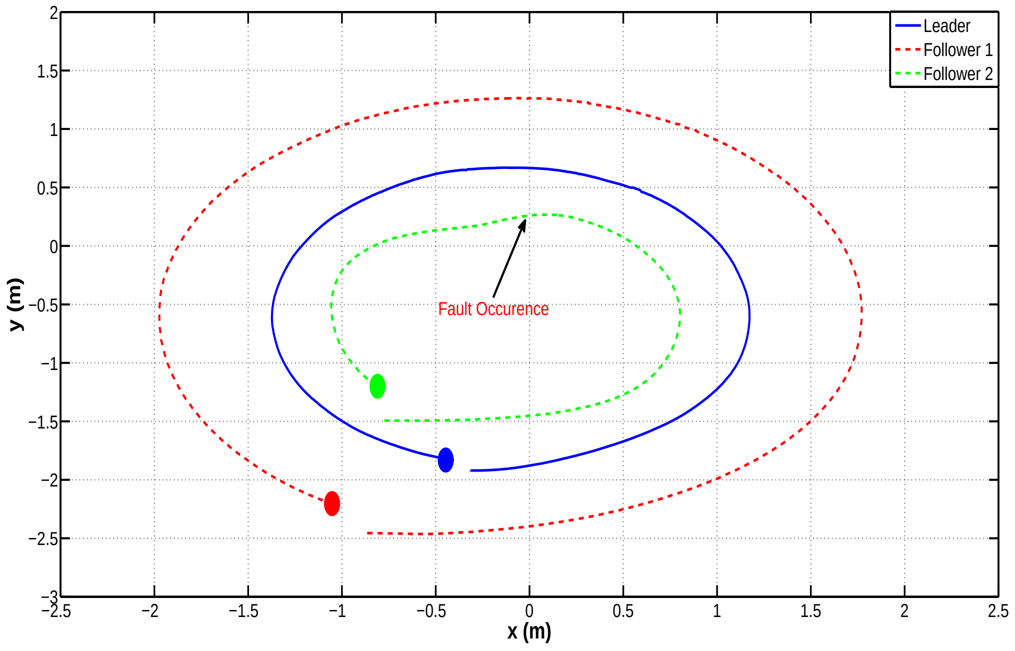


Figure 4.27: Robots' trajectories during mission execution considering the fault occurred in the second follower in Case 2

As can be observed from Figure 4.28, the proposed motion re-coordination is achieved. All the robots after 15 sec reduce the linear velocities by incorporating the fault occurred in the second follower. Each robot moves only with about 62% of its capability. If the healthy robots do not reduce their capabilities, then the desired formation configuration cannot be maintained due to the degraded capability of the faulty robot. Therefore, it reduces but with expected formation performance should be achieved in such a case.

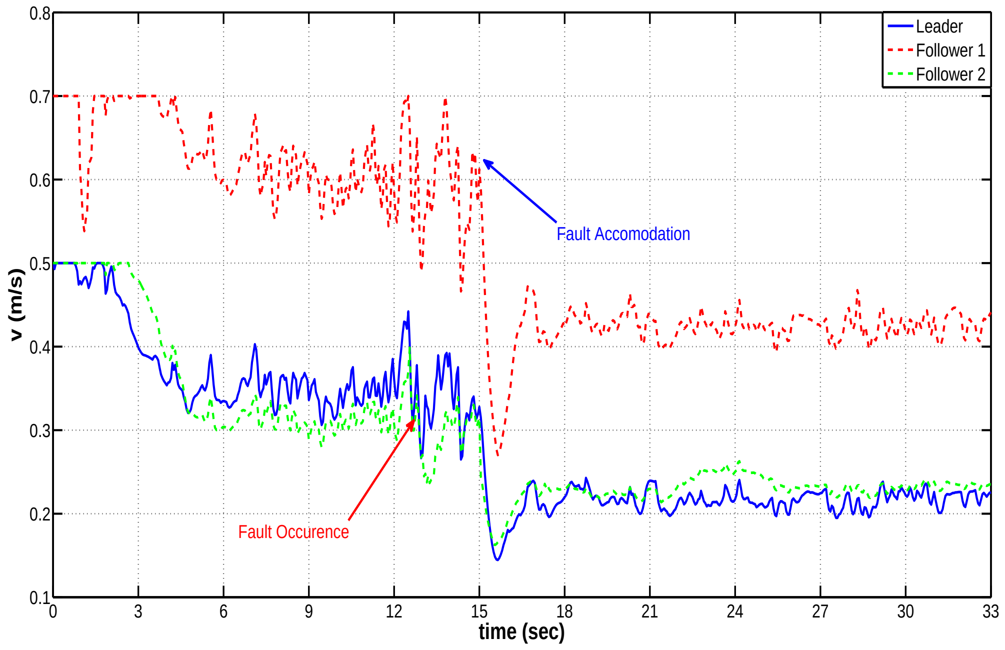


Figure 4.28: Robots' linear velocities during mission execution considering the fault occurred in the second follower in Case 2

Chapter 5

FTCC of Multiple WMRs Based on Particle Swarm Optimization (PSO)

In this chapter, an FTCC algorithm based on time optimal formation reconfiguration is presented. As mentioned in Chapter 4, there are two main challenges in case of severe fault occurrence; **i)** how to generate the new formation configuration, which is already solved by the optimal assignment (as presented in Section 4.2.1); and **ii)** how to reconfigure the formation to the new one.

As mentioned in Chapter 2, during formation reconfiguration, sudden changes have occurred to the robots' control inputs. If these inputs are not bounded, the robots behavior will become unstable, the formation may be broken, and the robots may collide. In Chapter 4, MPC is adopted as the WMRs' controllers. Then, the control inputs' constraints can be considered. In case of using any other controllers rather than MPC, the control inputs can't be bounded.

To solve this problem, formation reconfiguration stage is formulated as an optimal control problem considering dynamic and algebraic constraints, hence intelligence optimization methods can be utilized to obtain the optimal solution. The objective is to achieve the proposed formation configuration within minimum time considering the control inputs constraints and avoiding the collision among team members. A hybrid approach based on CPTD and PSO is adopted to solve this problem.

This chapter is organized as follows. First, the design of the nonlinear controller that achieves the desired formation in fault-free case is presented. Second, the proposed FTCC algorithm is illustrated. Finally, simulation and experimental results are presented to validate the effectiveness of the proposed algorithm.

5.1 Nonlinear Formation Controller in Fault-Free Case

The nonlinear controller presented here is similar to (Li et al., 2004) and (Tao and Shan, 2014). Let a team of j differentially-driven WMRs moving within a leader-follower scheme, where $j \in \{l, 1, 2, \ldots, N\}$ denotes the formation configuration of the leader l and N followers. As mentioned in Section 3.3, the leader l should track a predefined trajectory $(x_r(t), y_r(t))$ in a time interval $t \in [0, T]$. On the other hand, the followers N should follow the leader maintaining a desired formation configuration $(l_d - \phi_d)$ relative to the leader where l_d and ϕ are the follower's relative distance and angle with respect to the leader.

Let l_x and l_y be the projection of l_d along x and y coordinates, thus:

$$l_x = -(x_l - x_f)\cos\phi_l - (y_l - y_f)\sin\phi_l \tag{5.1}$$

$$l_y = (x_l - x_f)\sin\phi_l - (y_l - y_f)\cos\phi_l \tag{5.2}$$

Also, the desired formation configuration can be presented along the x and y axes as:

$$l_x^d = l_d\cos\phi_d \tag{5.3}$$

$$l_y^d = l_d\sin\phi_d \tag{5.4}$$

Differentiating Eqs. (5.1), (5.2), (5.3), and (5.4) achieves:

$$\begin{aligned}\dot{l}_x &= (x_l - x_f)\sin(\phi_l)\omega_l - (\dot{x}_l - \dot{x}_f)\cos\phi_l - (y_l - y_f)\cos(\phi_l)\omega_l - (\dot{y}_l - \dot{y}_f)\sin\phi_l \\ &= l_y\omega_l - \dot{x}_l\cos\phi_l - \dot{y}_l\sin\phi_l + \dot{x}_f\cos\phi_l + \dot{y}_f\sin\phi_l\end{aligned} \tag{5.5}$$

$$\begin{aligned}\dot{l}_y &= (x_l - x_f)\cos(\phi_l)\omega_l + (\dot{x}_l - \dot{x}_f)\sin\phi_l + (y_l - y_f)\sin(\phi_l)\omega_l - (\dot{y}_l - \dot{y}_f)\cos\phi_l \\ &= -l_x\omega_l + \dot{x}_l\sin\phi_l - \dot{y}_l\cos\phi_l - \dot{x}_f\sin\phi_l + \dot{y}_f\cos\phi_l\end{aligned} \tag{5.6}$$

From the kinematics of the differentially-driven WMR, it is known that:

$$v = \dot{x}\cos\phi + \dot{y}\sin\phi \tag{5.7}$$

and the nonholonomic constraint of the differentially-driven WMRs is presented as:

$$\dot{x}\sin\phi - \dot{y}\cos\phi = 0 \tag{5.8}$$

Using Eqs. (5.7) and (5.8), one can obtain:

$$\dot{l}_x = l_y\omega_l - v_l + \dot{x}_f\cos\phi_l + \dot{y}_f\sin\phi_l \tag{5.9}$$

$$\dot{l}_y = -l_x\omega_l - \dot{x}_f\sin\phi_l + \dot{y}_f\cos\phi_l \tag{5.10}$$

An error variable $\phi_e = \phi_f - \phi_l$ is defined to represent the difference between the leader and follower orientation angles. Substituting the error in Eqs. (5.9) and (5.10), and employing the trigonometric identities, then:

$$\begin{aligned}\dot{l}_x &= l_y\omega_l - v_l + \dot{x}_f\cos(\phi_f - \phi_e) + \dot{y}_f\sin(\phi_f - \phi_e)\\ &= l_y\omega_l - v_l + (\dot{x}_f\cos\phi_f + \dot{y}_f\sin\phi_f)\cos\phi_e + (\dot{x}_f\sin\phi_f - \dot{y}_f\cos\phi_f)\sin\phi_e\\ &= l_y\omega_l - v_l + v_f\cos\phi_e\end{aligned} \tag{5.11}$$

$$\begin{aligned}\dot{l}_y &= -l_x\omega_l - \dot{x}_f\sin(\phi_f - \phi_e) + \dot{y}_f\cos(\phi_f - \phi_e)\\ &= -l_x\omega_l - (\dot{x}_f\sin\phi_f - \dot{y}_f\cos\phi_f)\cos\phi_e + (\dot{x}_f\cos\phi_f - \dot{y}_f\sin\phi_f)\sin\phi_e\\ &= -l_x\omega_l + v_f\sin\phi_e\end{aligned} \tag{5.12}$$

To achieve the desired formation, it is required to design a control law to get v_f and ω_f to ensure that l_x and l_y converge to the desired values l_x^d and l_y^d, i.e.,

$$\lim_{t\to\infty}(l_x^d(t) - l_x(t)) = 0 \tag{5.13}$$

$$\lim_{t\to\infty}(l_y^d(t) - l_y(t)) = 0 \tag{5.14}$$

Considering the formation errors $x_e = l_x^d - l_x$ and $y_e = l_y^d - l_y$, then the control objective is to design a control law for x_e, y_e and ϕ_e asymptotically stable. Since the desired formation configuration $(l_d - \phi_d)$ is constant during the mission execution, $\dot{l}_x^d = \dot{l}_y^d = 0$. As a result, the error dynamics is written as:

$$\dot{x}_e = \dot{l}_x^d - \dot{l}_x = -l_y\omega_l + v_l - v_f \cos\phi \tag{5.15}$$

$$\dot{y}_e = \dot{l}_y^d - \dot{l}_y = l_x\omega_l - v_f \sin\phi \tag{5.16}$$

$$\dot{\phi}_e = \phi_f - \phi_l \tag{5.17}$$

When $l_y = l_y^d - y_e$ and $l_x = l_x^d - x_e$, then the error dynamics can be represented as:

$$\dot{x}_e = y_e\omega_l - l\sin\phi_d\omega_l - v_f\cos\phi_e + v_l \tag{5.18}$$

$$\dot{y}_e = -x_e\omega_l + l\cos\phi_d\omega_l - v_f\sin\phi_e \tag{5.19}$$

$$\dot{\phi}_e = \phi_f - \phi_l \tag{5.20}$$

Define new variables f_1 and f_2, where:

$$f_1 = -l\sin\phi_d\omega_l + v_l \tag{5.21}$$

$$f_2 = l\cos\phi_d\omega_l \tag{5.22}$$

then the error dynamics can be further written as:

$$\dot{x}_e = y_e\omega_l - v_f\cos\phi_e + f_1 \tag{5.23}$$

$$\dot{y}_e = -x_e\omega_l - v_f\sin\phi_e + f_2 \tag{5.24}$$

$$\dot{\phi}_e = \phi_f - \phi_l \tag{5.25}$$

Finally, the control law is designed as:

$$v_f = (k_1x_e + \omega_l y_e + f_1)\cos\phi_e - (-k_2y_e + \omega_l x_e - f_2)\sin\phi_e \tag{5.26}$$

$$\omega_f = (-k_1x_e - \omega_l y_e - f_1)\sin\phi_e - (-k_2y_e + \omega_l x_e - f_2)\cos\phi_e \tag{5.27}$$

where $k_1 > 0$ and $k_2 > 0$ are the controller gains.

5.2 FTCC Algorithm

In case of severe fault occurrence, the main purpose of the FTCC is to:

1. Detect and diagnose the actuator fault;
2. Re-assign the formation mission on the healthy members if one or more robots cannot complete the mission due to severe faults;
3. Reconfigure the formation according to the re-assigned mission within the robots' input constraints; and
4. Avoid the collision between the team members during reconfiguration.

As illustrated in Figure 5.1, the FTCC scheme consists of:

1. The FDD scheme in order to detect and diagnose the value of loss of effectiveness of actuator faults;
2. The task re-assignment and decision making mechanism in which the mission is re-assigned based on the new situation; and
3. The formation reconfiguration algorithm within constraints of the robots' control inputs and collision avoidance.

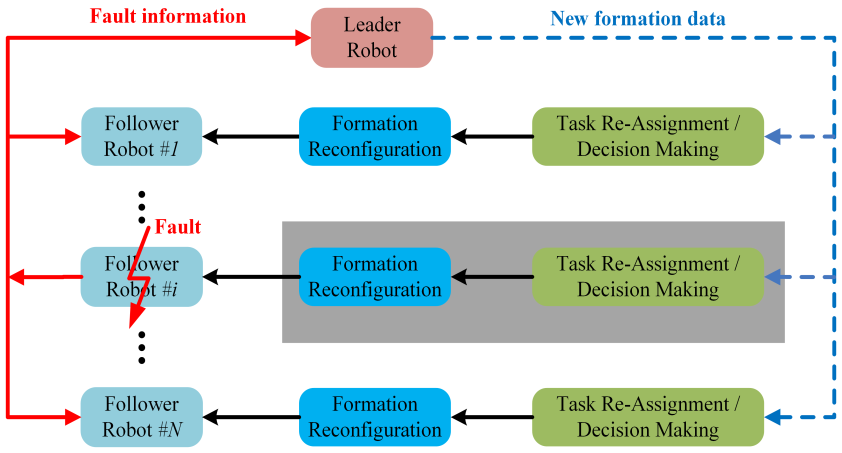

Fault information
Leader Robot
New formation data
Follower Robot #1
Formation Reconfiguration
Task Re-Assignment / Decision Making
Fault
Follower Robot #i
Formation Reconfiguration
Task Re-Assignment / Decision Making
Follower Robot #N
Formation Reconfiguration
Task Re-Assignment / Decision Making

Figure 5.1: Task re-assignment mechanism

Remark 5.1. *The* FDD *unit and the task re-assignment and decision making algorithm presented in Chapter 4 are applied to achieve the first and second objectives of the* FTCC *algorithm. Here, the formation reconfiguration algorithm considering the control inputs' and collision avoidance constraints will be presented.*

After deciding the optimal formation configuration (by means of the optimal assignment), the team members start reconfiguring their positions. The objective is to achieve the desired formation configuration in minimum time, considering the control inputs constraints and avoiding the collision during reconfiguration.

PSO is a population based stochastic optimization technique that mimic the behavior of a colony or swarm of insects, a flock of birds; or a school of fish. The PSO algorithm mimics the behavior of these social organisms. The word *particle* denotes, for example, a *bee* in a colony or a bird in a flock. Each particle in a swarm behaves in a distributed way using its own intelligence and the group intelligence of the swarm. As such, if one particle discovers a good path to food, the rest of the swarm will also be able to follow the good path instantly even if their location is far away in the swarm (Rao, 2009). The PSO algorithm was originally proposed by Eberhart and Kennedy (1995).

PSO algorithm initiates a random population of potential solutions to search an optimal solution within the optimization problem hyperspace (Arora, 2012). A single potential solution is called particle. Each particle p in the swarm is assumed to have two characteristics: a position and a velocity. Each particle keeps tracking its own current position and its best solution (in terms of the food source or objective function value) achieved during running the algorithm. According to the philosophy of PSO, each particle stores not only its current value but also its best value achieved so far. This best value is called the local best and is denoted by x_p. In addition, each particle tracks the best position for the entire swarm, which is called the global best and denoted by x_G. PSO algorithm consists of changing the velocity of each particle in each iteration towards x_p and x_G (Arora, 2012).

Compared to other stochastic optimization technique such as GA, PSO has an attractive feature is that it has fewer algorithmic parameters to specify and adjust. It does not use any of the GAs' evolutionary operators such as crossover and mutation. Also, the algorithm does not require binary number encoding or decoding. Thus, it is easier to implement on the computer. Moreover, PSO can find the optimal solution with a fast convergence speed. PSO has been successfully applied to many classes of problems, such as mechanical and structural optimization and multi-objective optimization, artificial neural network training, and fuzzy system control (Arora, 2012).

Recalling Section 2.3.1, the robots' control inputs are continuous. However, PSO cannot solve the objective function with continuous control inputs. In this case, the control inputs should be piecewise linearized to replace the continuous ones. CPTD method is thereby adopted to obtain the approximate objective function and the constraints to simplify the problem for the PSO.

5.2.1 Problem Formulation

As mentioned in Section 2.3.1, the posture of each healthy robot is defined as:

$$q_i = [x_i, y_i, \phi_i]^T \in \mathbb{R}^3, \quad \forall i \in \{1, 2, \dots, F\}.$$

where F represents the number of the remaining healthy robots.

The mathematical model of the robot presented in Eq. (2.4) can be represented in the following compact form:

$$\dot{q}_i(t) = f(t, q_i(t), u_i(t)), t \in [0, T], \forall i \in \{1, 2, \dots, F\}, \tag{5.28}$$

where T is time of reconfiguration. Therefore, for the whole team of healthy robots, define the state and control inputs vectors as:

$$X = [q_1^T, q_2^T, \dots, q_F^T] \in \mathbb{R}^{3F},$$
$$U = [u_1, u_2, \dots, u_F] = \{U(t) | \forall t \in [0, T]\},$$

and the formation system dynamics can be described as:

$$\dot{X}(t) = f(t, X(t), U(t)), t \in [0, T]. \tag{5.29}$$

Given the continuous control inputs U and the initial state $X(0) = X_0$, the state of the whole system at any time $t \in [0, T]$ can be determined uniquely in the following form:

$$X(t) = X(0) + \int_0^{t^-} f(\tau, X(\tau), U(\tau))d\tau. \tag{5.30}$$

According to Eq. (5.30), the state $X(t)$ can be defined only by the control inputs U in the form of $X(t|U)$.

5.2.2 Objective Function and Constraints

According to (Furukawa et al., 2003), the standard canonical form of the objective function can be expressed as:

$$J(U) = \Phi_0(X(T|U)) + \int_0^{T} L_0(t, X(t|U), U(t))dt, \tag{5.31}$$

and the general form of the M equality and inequality constraints is given by:

$$g_i(U) = \Phi_i(X(\tau_i|U)) + \int_0^{\tau_i} L_i(t, X(t|U), U(t))dt \leq 0, \forall i \in \{1, \dots, M\}. \tag{5.32}$$

The time optimal control problem for the whole formation can be formulated as finding the optimum control inputs U and the terminal time T to minimize the objective function $J(U)$ such that:

$$\min_{U,T} J(U), \tag{5.33}$$

$$J(U) = T, \tag{5.34}$$

subject to the following constraints:

(i) The primary constraints:

$$U_{min} \leq U(t) \leq U_{max}, \forall t \in [0, T], \tag{5.35}$$

$$T > 0. \tag{5.36}$$

(ii) The free terminal constraints, which means that at T, the team should reach the desired formation configuration. This constraint can be represented as:

$$g_1(U,T) = \sum_{i=2}^{F} \{q_i - q_1 - F_{dnew}\} = 0, \forall i = \{2, \ldots, F\}, \tag{5.37}$$

where F_{dnew} is the new desired formation between the leader robot denoted by 1, and the followers denoted by $\{2, \ldots, F\}$.

(iii) The collision avoidance constraint: The distance between any two robots i and j can be defined as:

$$d_{i,j}(t) = \sqrt{(x_i(t) - x_j(t))^2 + (y_i(t) - y_j(t))^2}. \tag{5.38}$$

To avoid the collision, $d_{i,j}(t)$ must be greater than the safety collision distance D, then the collision avoidance constraint can be presented as:

$$d_{i,j}(t) \geq D,\ \forall t \in [0,T],\ \forall i,j \in \{1, \ldots, F\},\ i \neq j. \tag{5.39}$$

5.2.3 CPTD Method

The idea of CPTD is to approximate the control inputs u_i by a piecewise function of a set of static parameters. The optimum time spectrum T is partitioned into n time intervals, and the time of each interval in Δt. The sets of static parameters and time-step interval are considered as the design variables of the optimization problem, while the objective function is minimized by PSO. The detailed procedure the steps of CPTD are:

Step 1 Dividing the time T: The optimum time T is divided into n intervals, $n \in \{1, 2, \ldots\}$ intervals, each of time $\Delta t \in \mathbb{R}^+$, thus,

$$T = n\Delta t. \tag{5.40}$$

According to the corresponding control inputs at each time interval, the states of the system are found by integrating Eq. (2.4).

Step 2 The piecewise linearization of the m control inputs: For the n time intervals, define $m \times n$ constants for the i^{th} healthy robot as:

$$\Omega_i = \{\sigma_j^i \in \mathbb{R}^m | \forall j \in \{1, \ldots, n\}\}, \forall i \in \{1, \ldots, F\}. \tag{5.41}$$

Then, each continuous control inputs for the i^{th} robot can be approximated by a piecewise function as:

$$\hat{u}_i(t; n, \Omega_i) = \sum_{j=1}^{n} \sigma_j \lambda_j(t) \equiv u_i(t), \tag{5.42}$$

where $\lambda_j(t)$ is given by:

$$\lambda_j(t) = \begin{cases} 1 & (j-1)\Delta t \le t \le j\Delta t \\ 0 & \text{otherwise.} \end{cases} \tag{5.43}$$

Eq. (5.43) guarantees that the robots stop when they reach the desired formation configuration.

Define the set of all piecewise constants for all healthy robots as $\Omega = \{\Omega_1, \ldots, \Omega_F\}$. Then, the set of approximated control inputs can be expressed as:

$$\hat{U}(t; n, \Omega) = \{\hat{u}_1(t, n, \Omega_1), \ldots, \hat{u}_F(t; n, \Omega_F)\}. \tag{5.44}$$

The objective turns to obtain the parameter set Ω. In real-time implementation, it is important to choose an appropriate value for n. Increasing n will exponentially increase the computational time, while reducing n will cause loss of accuracy (Furukawa et al., 2003).

Step 3 Re-formulating the objective function and constraints: Since Ω and Δt become the new design variables which replace the original ones u and T, the objective function and its constraints should be re-formulated to fit for the following optimization problem:

$$J \equiv \min_{\Omega, \Delta t} n\Delta t, \tag{5.45}$$

subject to:

$$0 < \Delta t, \tag{5.46}$$

$$(u_{min})_i \le \sigma_j^i \le (u_{max})_i, \forall j \in \{1, \ldots, n\}, \forall i \in \{1, \ldots, F\}, \tag{5.47}$$

$$\hat{g}_1(\Omega, \Delta t) = \sum_{i=2}^{F} \{q_i - q_1 - F_{dnew}\} = 0 \tag{5.48}$$

$$d_{i,j}(\Delta t) \ge D, \ \forall i, j \in \{1, \ldots, F\}, \ i \ne j. \tag{5.49}$$

Numerically, this problem can be regarded as a nonlinear constrained problem, in which PSO algorithm is applied. Figure 5.2 illustrates the proposed formation reconfiguration algorithm.

5.2.4 PSO Algorithm

Based on the above description, the objective is to obtain the optimum values of σ_j^i and Δt. The PSO can be executed as the following steps:

Step 1 Construction of the vector of particles' positions: For each robot i, the set of $m \times n$ control parameters Ω_i is

$$\Omega_i = [\sigma_{11}\ \sigma_{21}\ \ldots\ \sigma_{m1}\ \sigma_{12}\ \sigma_{22}\ \ldots\ \sigma_{m2} \ldots\ \ldots\ \sigma_{1n}\ \sigma_{2n}\ \ldots\ \sigma_{mn}]$$

Thus, the particle position including the time Δt can be expressed as $x = [\Omega_1\ \Omega_2\ \ldots\ \Omega_F\ \Delta t]$, where the length of x is $(N_{dv} = m \times n \times F + 1)$. The particles' position vector therefore can be written as:

$$x = [\sigma_{11}^1\ \sigma_{21}^1\ \ldots\ \sigma_{m1}^1\ \ldots\ \sigma_{1n}^1\ \sigma_{2n}^1\ \ldots\ \sigma_{mn}^1\ \sigma_{11}^2 \sigma_{21}^2\ \ldots\ \sigma_{m1}^2\ \ldots\ \sigma_{1n}^2\ \sigma_{2n}^2\ \ldots\ \sigma_{mn}^2 \ldots$$
$$\ldots\ \sigma_{11}^F\ \sigma_{21}^F\ \ldots\ \sigma_{m1}^F\ \ldots\ \sigma_{1n}^F\ \sigma_{2n}^F\ \ldots\ \sigma_{mn}^F\ \Delta t]$$

Step 2 Initialization: Select the number of particles in swarm N_p, the cognitive and social parameters c_1, c_2 respectively. Usually, $(N_p = 5N_{dv}$ to $10N_{dv})$, c_1, c_2 can be chosen between 0 and 4. Select the maximum number of iterations k_{max}. Set the initial velocity of each

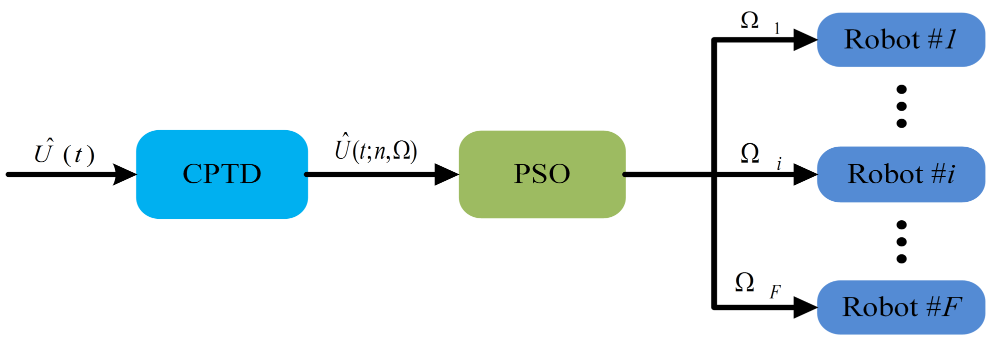


Figure 5.2: Time optimal formation reconfiguration controller block diagram

particle $v^{(i,0)}$ to 0. Set the iteration Counter k to 1. Using a random procedure, an initial random population of particles is generated based on the following:

$$x_j^{(i,0)} = X_{j_L} + r_{ij}(X_{j_U} - X_{j_L}), \forall j = \{1, \ldots, N_{dv}\}, \forall i = \{1, \ldots, N_p\}, \tag{5.50}$$

where r_{ij} is a uniformly distributed random number between 0 and 1. r_{ij} is generated for each design variable in each particle. X_{j_L} and X_{j_U} represent the lower and upper bounds of each design variable, respectively.

Step 3 Cost function calculation: To use PSO, the problem should be solved as an unconstrained problem. Therefore all constraints presented in Eqs. (5.35) to (5.39) should be included in the objective function. The objective function is formulated as:

$$J = \min_{\Omega, \Delta t} n\Delta t + \rho \cdot \hat{g}_1(\Omega, \Delta t) + \sum_{i=1}^{F-1} \sum_{j=i+1}^{F} [\rho_{ij} \cdot \max(0, D - d_{i,j}(t))], \tag{5.51}$$

where ρ and ρ_{ij} are the punishment constant coefficients. After calculating J, the local best solution for each particle x_p and the global best solution x_G are evaluated.

Step 4 Velocity calculation: Calculate the velocity of each particle as:

$$v^{(i,k+1)} = \omega v^{(i,k)} + c_1 r_1 (x_p^{(i,k)} - x^{(i,k)}) + c_2 r_2 (x_G^{(k)} - x^{(i,k)}), \forall i = \{1, \ldots, N_p\}, \tag{5.52}$$

where ω is the inertia weight which enables the swarm to converge more accurately and efficiently (Rao, 2009). ω can be calculated as:

$$\omega_k = \left(\omega_{max} - \frac{\omega_{max} - \omega_{min}}{k_{max}}\right) k, \tag{5.53}$$

where ω_{min} and ω_{max} are the initial and final values of the inertia weight with a commonly used values 0.4 and 0.9, respectively.

Step 5 Update the new positions of particles : The new position of each particles is:

$$x^{(i,k+1)} = x^{(i,k)} + v^{(i,k+1)}, \forall i = \{1, \ldots, N_p\}. \tag{5.54}$$

Step 6 Update the solution: Calculate the objective function at all new positions.

Step 7 Terminate criterion: Check for convergence of the iterative process. If the stopping criterion is satisfied, stop. Otherwise, set $k = k + 1$ and go to step 3.

The PSO procedure is illustrated in Figure 5.3, while the overall FTCC strategy is summarized in Algorithm 5.1.

Remark 5.2. *In this thesis, the stopping criterion is selected such that if the cost function does not change in 20 consecutive iterations, then terminate the iteration process.*

Algorithm 5.1 Task re-assignment and formation reconfiguration strategy for a team of WMRs

```
1:  for each follower do
2:      detect detect $\gamma^{j_i}$
3:      send detect $\gamma^{j_i}$ to the leader
4:      if $\bar{\gamma} \leq \gamma^{j_i} \leq 1$ then
5:          for the leader do
6:              determine the remaining number of the healthy robots $F = N - 1$
7:              send the new formation data $S$ to the followers
8:          end for
9:          for each healthy follower do
10:             calculate the cost matrix $c$
11:             apply the Hungarian algorithm
12:             assign to the corresponding slot
13:         end for
14:     end if
15: end for
16: Start the PSO procedure
17: Obtain the optimum solution
18: for each healthy robot do
19:     receive the optimum control inputs
20:     integrate Eq. (2.4)
21:     perform the formation reconfiguration
22: end for
```

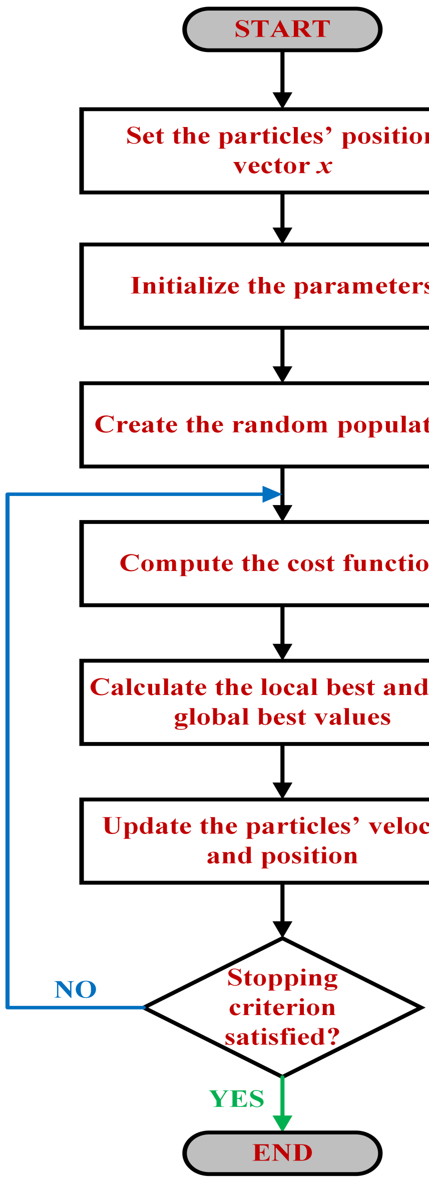

START
Set the particles' position vector *x*
Initialize the parameters
Create the random population
Compute the cost function
Calculate the local best and the global best values
Update the particles' velocity and position
Stopping criterion satisfied?
NO
YES
END

Figure 5.3: PSO flow chart

5.3 Numerical Validation via Simulation

To verify the proposed control strategies in this chapter, a group of WMRs are used in simulation to validate the effectiveness of the proposed control algorithm. All the values adopted for the controller parameters are shown in Table 5.1.

During simulation, a team of five WMRs perform a triangular formation with the same scenario presented in Section 4.3.2.

According to the assignment results, the healthy robots start reconfiguring their positions

Table 5.1: Control system parameters for the FTCC algorithm based on PSO

Parameter	Value	Parameter	Value
ω_{max}	0.5 rad/s	ω_{min}	-0.5 rad/s
v_{min}	0	v_{max}	0.5 m/s
c_1	2	c_2	2
k_{max}	2000	N_p	300
D	0.5 m		

to achieve the new formation. The proposed CPTD-PSO algorithm is capable of generating the optimum control inputs. Figures 5.4 and 5.5 illustrate the robots' trajectories during mission execution. On the basis of the Hungarian Algorithm, the 3^{rd} and the 4^{th} followers start changing their positions with respect to the leader to change the whole formation shape to be a diamond one. Figure 5.4 shows the effectiveness of proposed algorithm in both fault-free and faulty case. It's clear that the robots converge to the desired trajectories in the fault-free case, and healthy robots change their positions according to the results obtained from the Hungarian algorithm and the integrated CPTD-PSO approach. The ($\diamond$) marker indicates the position of the robots just before the formation reconfiguration. Figure 5.5 gives snapshots of the formation of the team during mission execution indicating the fault-free, reconfiguration, and after fault occurrence stages.

Figure 5.6 illustrates the robots' trajectories during the reconfiguration stage only. Within a maximum distance of 4 m, each robot reaches the final formation configuration. Also. it is evident that during reconfiguration the robots achieve the collision avoidance constraint presented in Eq. (5.39).

Figures 5.7 and 5.8 exhibit the optimal control inputs for the healthy robots during reconfiguration. It is observed that the PSO algorithm can satisfy the control inputs' constraints presented in Eq. (5.35). Also, it is shown that the time of each interval Δt is almost 2.75 sec. As can be seen from Figure 5.9, the distances between all robots including the faulty one are greater than the safety distance D meaning that the proposed algorithm achieves the collision avoidance constraint presented in Eq. (5.39). It is revealed in Figure 5.10 the objective function converges to the minimum value within a 457 iterations. Note that the computational time of the algorithm consumes only 15 sec.

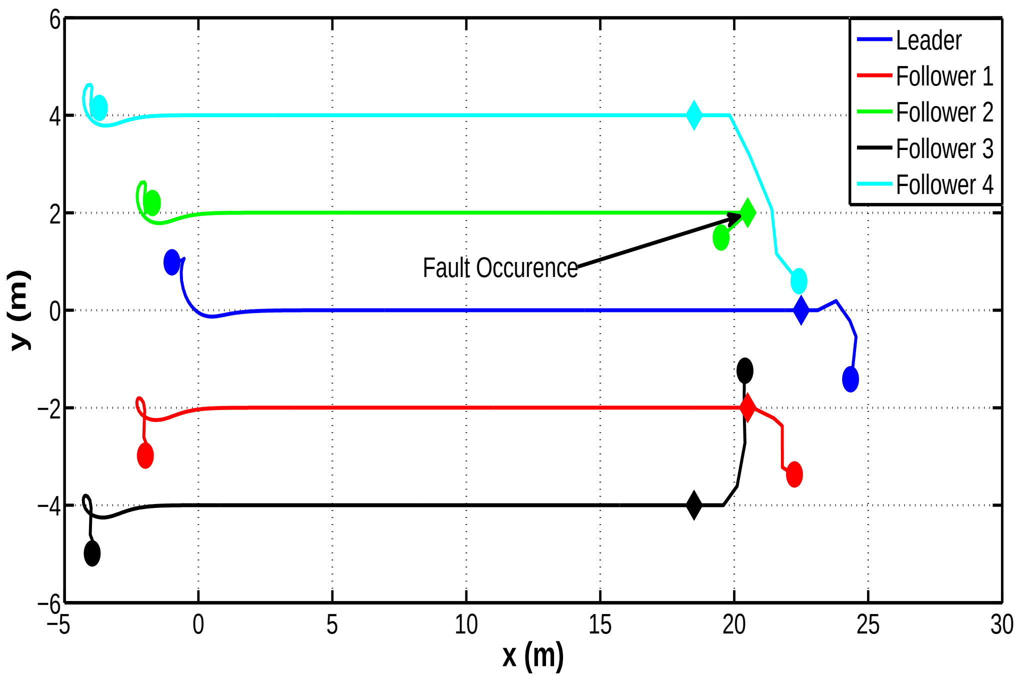


Figure 5.4: Robots' trajectories during mission execution in simulation

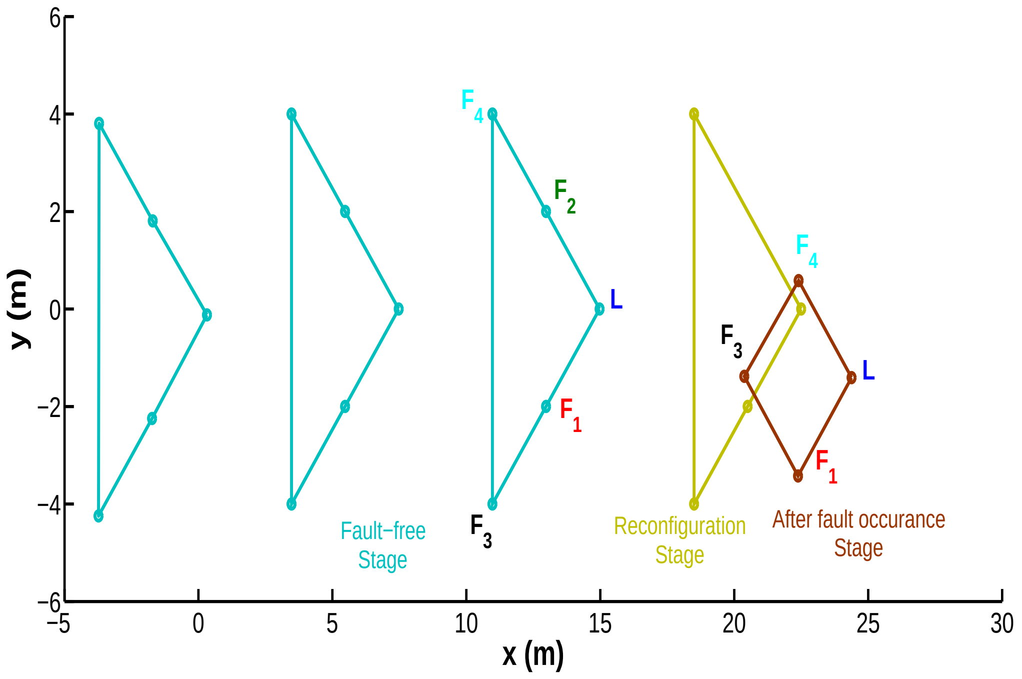


Figure 5.5: Snapshot of the formation of the team during simulation

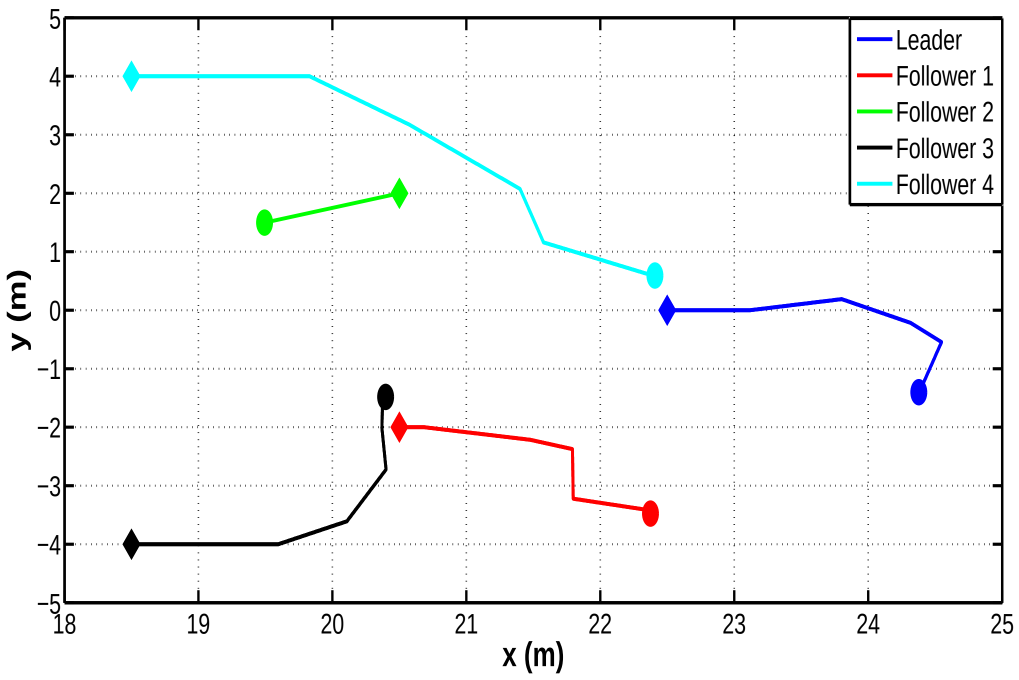


Figure 5.6: Robots' trajectories during formation reconfiguration during simulation

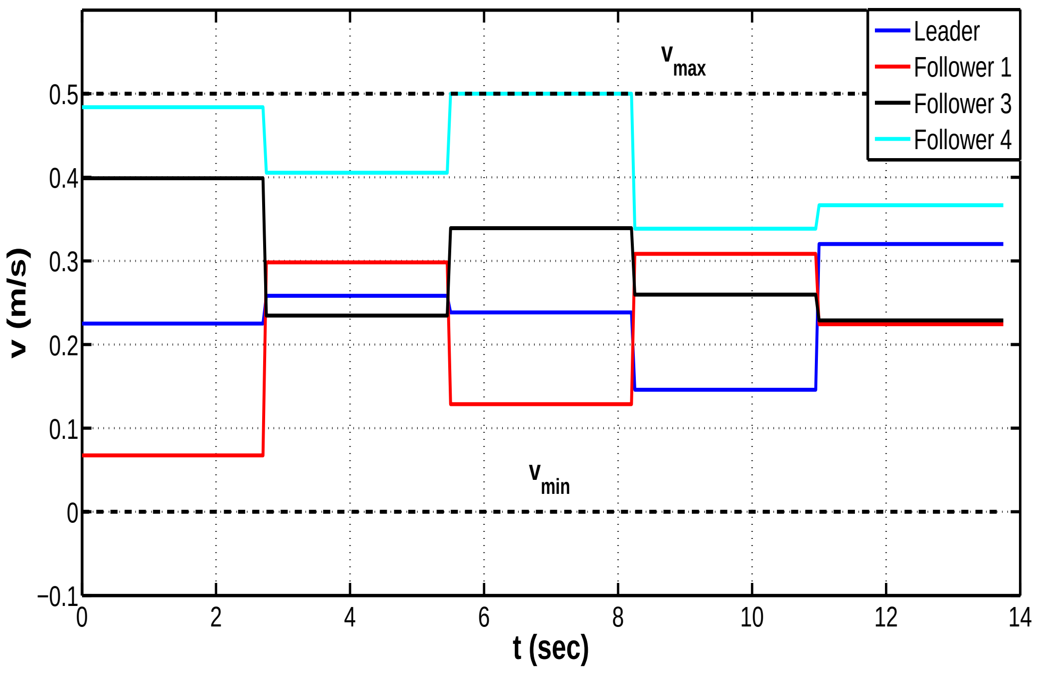


Figure 5.7: Robots' linear velocities during reconfiguration in simulation

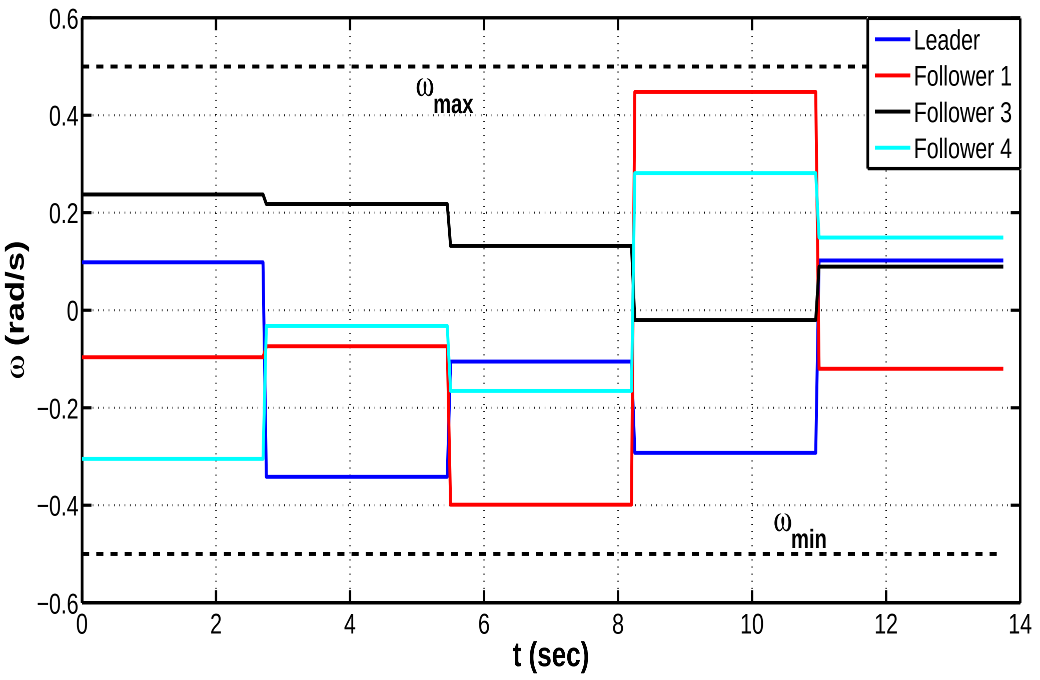


Figure 5.8: Robots' angular velocities during reconfiguration

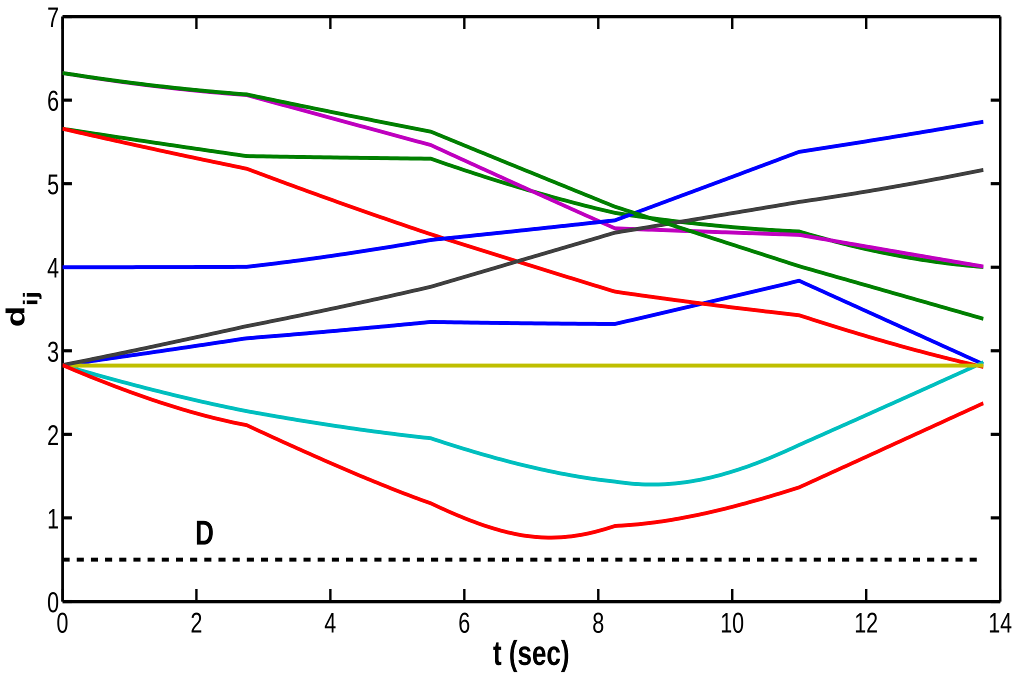


Figure 5.9: Distances between robots during reconfiguration in simulation

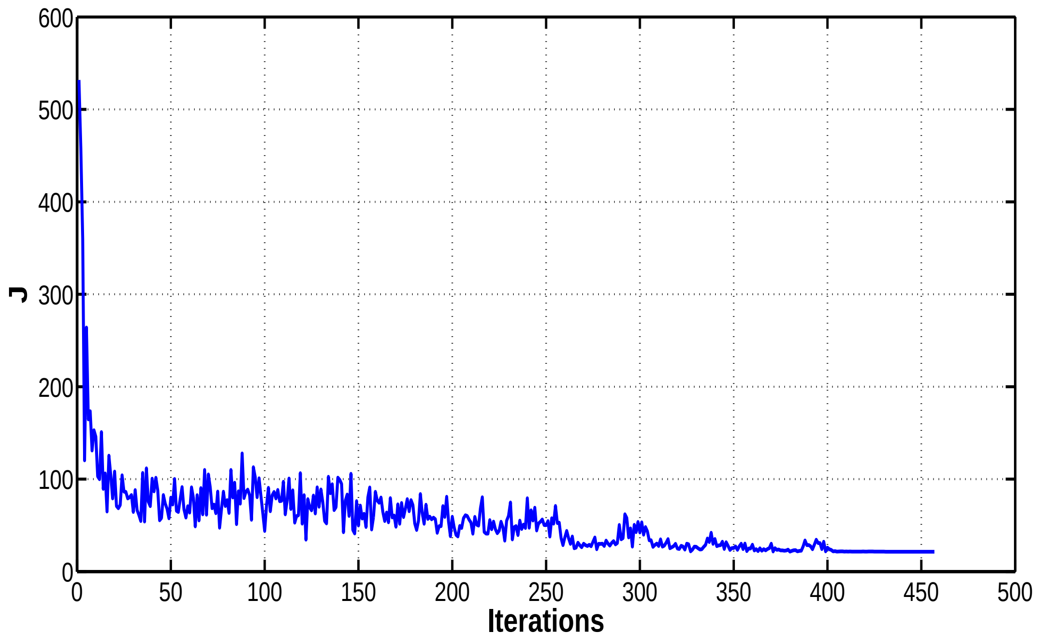


Figure 5.10: The objective function history along the number of iterations during simulation

5.4 Experimental Results Assessment

Due to the space limitation in the laboratory, and since the FDD scheme and the task re-assignment algorithm are already validated in Chapter 4, only three robots are used in the experiment considering the formation reconfiguration stage only. i.e., it is assumed that the team consists of four robots, while one of them (the virtual one) subject to severe fault, then the other three robots start reconfigure their formation. Therefore the experiment started from the reconfiguration stage with three robots as shown in Figure 5.11.

Once the first follower (the virtual one) is subject to a severe fault, it separated from the formation due to the actuator fault. the leader sends new formation commands to the

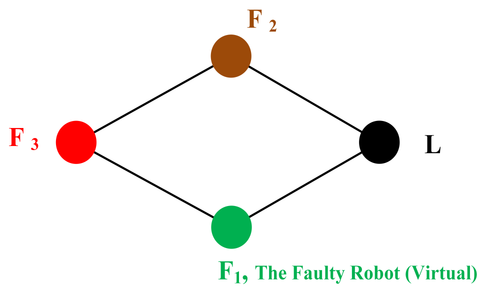


Figure 5.11: The configuration of robots during the experiment

remaining three followers as slots' coordinates with respect to the leader position. These data are:

$$S_1 = [0.65, -0.65], \qquad S_2 = [-0.65, -0.65].$$

According to the Hungarian algorithm, the second follower remains in its position, while the third one changes its position to fill the first slot.

According to the assignment results, the three healthy robots start reconfiguring their positions to achieve the new formation. The optimal control inputs are generated by the developed algorithm. Figures 5.12 and 5.13 illustrate the robots' trajectories during reconfiguration. Based on the assignment results, third follower starts change its position relative to the leader such that the whole formation shaped can be maintained. It is evidenced that within a maximum distance of 2 m, each robot reaches the final formation configuration.

Figures 5.14 and 5.15 show the optimal control inputs for the robots during reconfiguration. The CPTD-PSO algorithm can satisfy the control inputs' constraints presented in Eq. (5.35). Also, the time of each interval Δt is almost 2 sec. As can be seen from Figure 5.16, the distances between all robots are greater than the safety distance D achieving the collision avoidance constraint in Eq. (5.39).

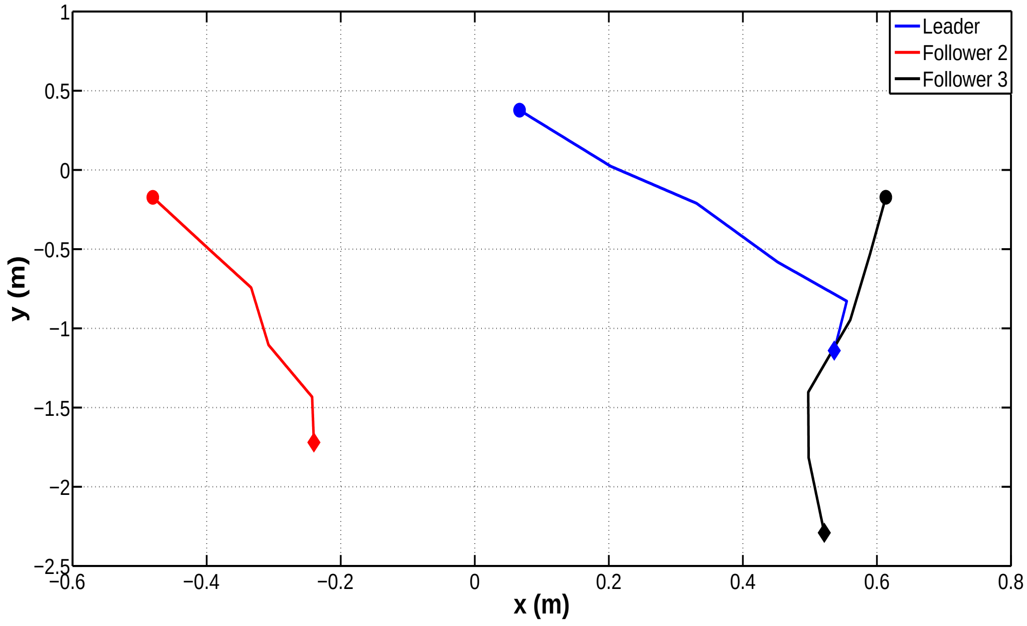


Figure 5.12: Robots' trajectories during the experiment

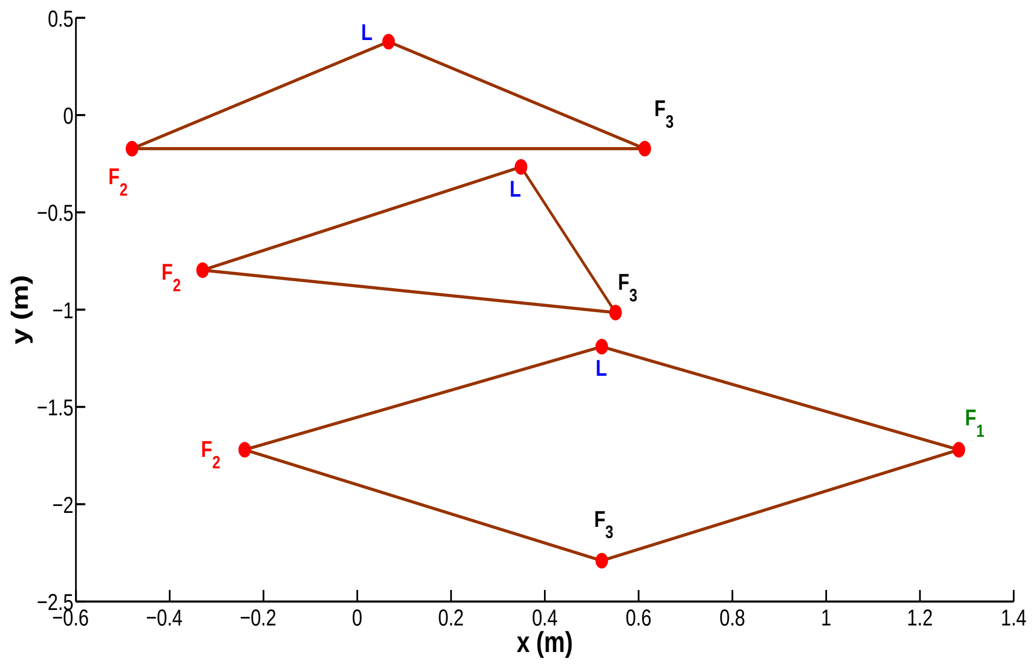


Figure 5.13: Snapshot of the formation of the team during the experiment

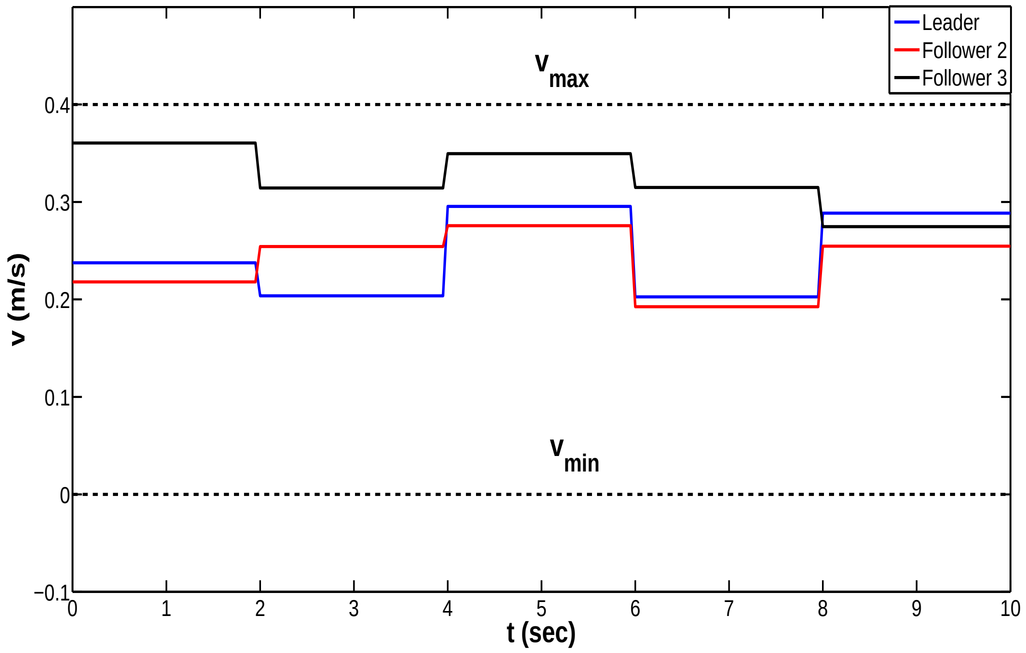


Figure 5.14: Robots' linear velocities during the experiment

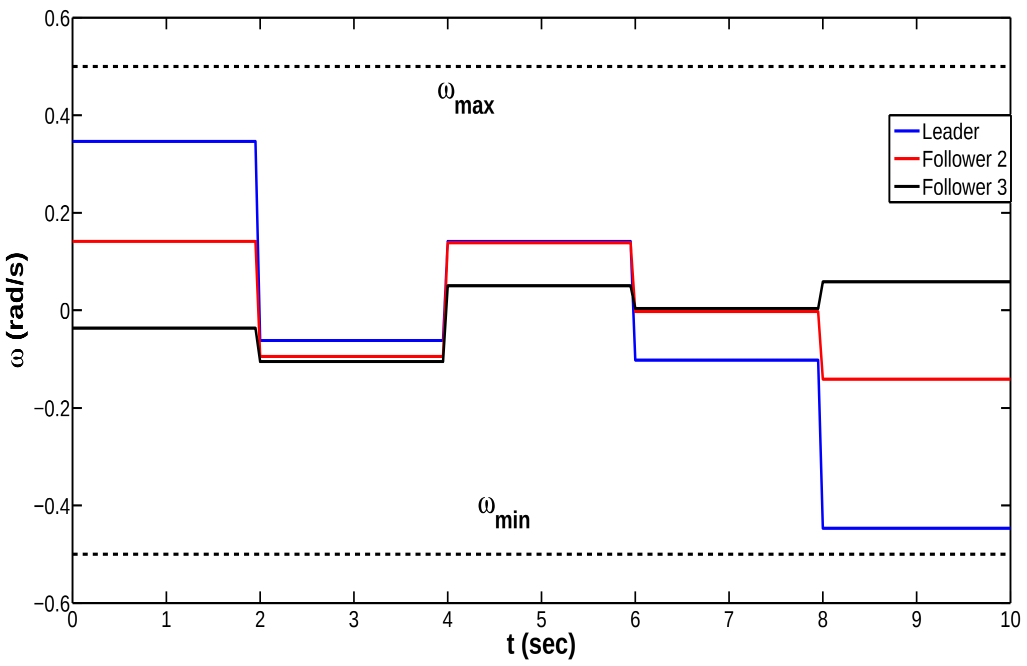


Figure 5.15: Robots' angular velocities during the experiment

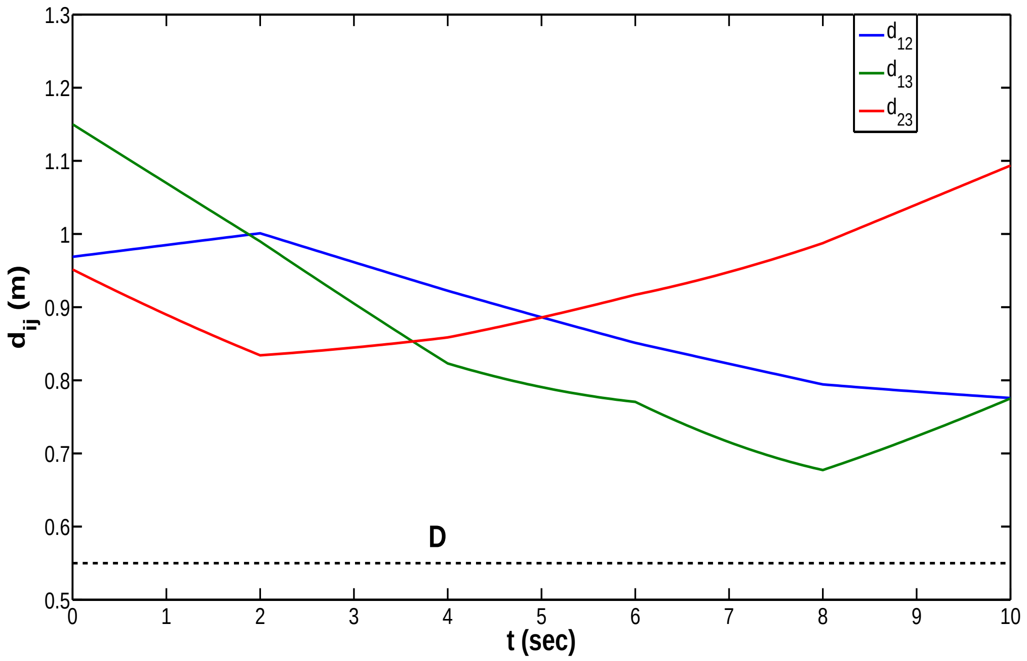
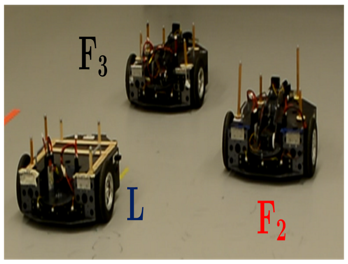
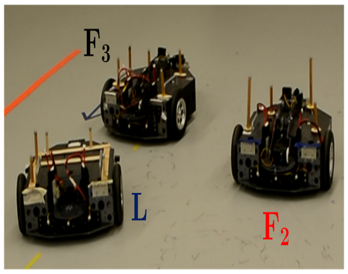


Figure 5.16: Distances between robots during the experiment

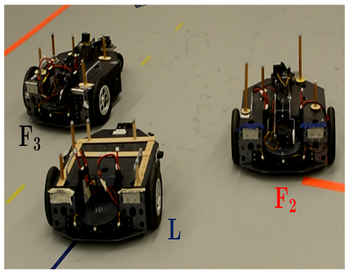
Figure 5.17 illustrates snapshots of the real experiment. Initially, when $t = 1$ sec, the robots are in their positions. at $t = 6$ sec, the robots are still moving to achieve the new configuration; when $t = 10$ sec, the robots have successfully completed the desired formation configuration.



(a) $t = 1\ s$



(b) $t = 9\ s$



(c) $t = 10\ s$

Figure 5.17: Snapshots of the experiment

Chapter 6

Conclusions and Future Works

6.1 Conclusions

In this dissertation, cooperative control strategies of WMRs in normal and faulty cases are presented. Different algorithms are proposed step by step to address some drawbacks of the preceding algorithms. These algorithms are verified in numerical simulations and real-time experiments, while more emphasis has been placed on experimentally verifying the developed approaches through indoor testing. The successful implementation of the proposed algorithms is considered as a key aspect of the overall contribution.

In Chapter 3, the problems of trajectory tracking, as well as formation control of multiple WMRs are addressed. A novel algorithm combining both dynamic feedback linearization and LMPC is presented. The advantage of the developed algorithm is that by using feedback linearization, a nonlinear problem is solved and the robot model becomes LTI with new control inputs. Moreover, LMPC is applied to the linearized model such that the optimum control inputs are generated to perform the trajectory tracking as well as the entire formation mission. Based on such a control design strategy, the computational burden associated with MPC (especially with increasing the number of robots) can be significantly reduced since the model is LTI. The proposed algorithm is accompanied by a theoretical proof of stability. Compared to the relevant works, this algorithm can be applied in real-time since the computational burden issue has been avoided. An obstacle avoidance algorithm based on the mechanical impedance concept is adopted to each robot controller, allowing working in cluttered environments. To prove the applicability of the proposed control algorithm in real applications, a team of WMRs is paired with a team of UAVs for forest monitoring and fire detection application. In this scheme, the problem of limited running time and limited payload of UAVs can be avoided. Another advantage of this algorithm lies in that communication efforts between the UAVs and the ground station can be decreased, since the WMR leader is identified as the local mobile ground station during gathering data about the fire evolution. This saves more UAVs power, and allows rapid analysis of the data collected by the UAVs. Finally, the proposed control policies are validated by presenting the simulations results and experimental results in different cases.

FTCC of WMRs is investigated in Chapter 4. Compared to the existing studies in the literature, this work provides a new solution for FTCC problem with integration of the FDD function and experimental testing validation. FDD of actuator faults in the team has been achieved by the TSKF algorithm and integrated with the FTCC strategy to form an active FTCC framework. The proposed FTCC scheme is capable of **i)** reconfiguring the formation if one or more robots are faulty and cannot complete the mission; and **ii)** re-coordinating the motion of each robot in the team if one or more robots subject to fault but can still continue the mission with degraded performance. Furthermore, both the numerical and experimental results exemplify that the formation system is stabilized, converges to the desired formation and reconfigures the formation shape in the presence of severe actuator faults, and re-coordinate the mission in the event of non-severe faults. In case of severe fault occurrence, two methods are used. The Graph Theory approach, and the optimal assignment. Although the FTCC algorithm based on the Graph Theory is simple, decentralized, and can deal with the case of the faulty leader since all robots have a *prior* knowledge of all possible fault cases. However, it is difficult to implement in real-time especially with increasing the number of robots. On the other hand, formulating the FTCC problem as an optimal assignment can solve the previously mentioned demerits.

In Chapter 5, the problem of FTCC is further studied. Based on a combination of CPTD and PSO, an FTCC scheme is proposed to reconfigure the formation when one or more robots cannot complete the mission due to severe faults. The proposed algorithm has three major advantages: i) it is simple and can be implemented in real-time; ii) there is no need to use a specific controller such as MPC to handle the robots' constraints since the reconfiguration problem is formulated as a constrained optimization problem; and iii) the collision avoidance is explicitly considered in the design procedure. Therefore, there is no need to add a collision avoidance algorithm, resulting in less complexity of the robot control system. Moreover, any other constraint can be directly added to the problem according to the mission specifications. From the aspect of computational burden, it is favorable to apply the proposed algorithm in real-time applications since PSO converges rapidly to the optimum solution. Simulation and experimental results have validated the applicability of the proposed scheme.

6.2 Future Works

Despite various degrees of success have been achieved in this thesis work, cooperative control of WMRs is still very challenging. Since optimal control strategies like MPC is highly desired in multi-vehicle cooperation, then learning-based model predictive control (LBMPC) is considered as a promising approach. Most recently, LBMPC has been applied to a team of UAVs but not applied yet to the WMRs. LBMPC combines techniques from statistical learning, which provides an improvement in the performance, with control tools which guarantee stability, robustness, and safety. The idea is that the robots can learn their unmodeled dynamics using the learning algorithm at every time step and the updated system is controlled by the MPC to achieve the desired formation configuration.

In this dissertation, only robot kinematics are considered. With increasing the robots' weight and speed, robot dynamics cannot be ignored. Designing MPC controllers to handle robot dynamics to form a two-layer MPC approach could be further investigated.

More studies on FTCC are still needed. Sensor and the communication faults/failures are not considered in this thesis, although they are very important from the aspect of system's safety. Different decision making algorithms rather than the optimal assignment can be used. For example, fuzzy logic can be considered as an excellent option. Moreover, the case of the faulty leader needs further investigation.

6.3 Publications

6.3.1 Journal Publications

Kamel, M. A., Yu, X., and Zhang, Y. M. (2016). Fault-tolerant cooperative control design of multiple wheeled mobile robots. Submitted to *IEEE Transactions on Control Systems Technology* (Under review).

6.3.2 Conference Publications

Kamel, M. A., Yu, X., and Zhang, Y. M. (2016). Fault-tolerant cooperative control of WMRs under actuator faults based on particle swarm optimization. In *International Conference on Control and Fault-Tolerant Systems (SysTol)* (accepted and to be presented).

Kamel, M. A., Yu, X., and Zhang, Y. M. (2016). Design of fault-tolerant cooperative control algorithm applied to WMRs against actuator faults. In *American Control Conference (ACC)*, pages 7092–7097.

Ghamry, K. A., Dong, Y., **Kamel, M. A.**, and Zhang, Y. M. (2016). Real-time autonomous take-off, tracking and landing of UAV on a moving UGV platform. In *Mediterranean Conference on Control and Automation (MED)*, pages 1236–1241.

Kamel, M. A., Yu, X. , and Zhang, Y. M. (2016). Real-time optimal formation reconfiguration of multiple wheeled mobile robots based on particle swarm optimization. In *World Congress on Intelligent Control and Automation (WCICA)*, pages 703–708.

Kamel, M. A., Ghamry, K. A., and Zhang, Y. M. (2016). Real-time Fault-tolerant cooperative control of multiple UAVs-UGVs in The presence of actuator faults. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1267–1272.

Hafez, A. T. and **Kamel, M. A.** (2016). Fault-tolerant control for cooperative unmanned aerial vehicles formation via fuzzy logic. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1261–1266.

Ghamry, K. A., **Kamel, M. A.**, and Zhang, Y. M. (2016). Cooperative forest monitoring and fire detection using a team of UAVs-UGVs. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1206–1211.

Kamel, M. A., Zhang, Y. M., and Yu, X. (2015). Fault-tolerant cooperative control of multiple wheeled mobile robots under actuator faults. In *IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS)*, pages 1152–1157.

Kamel, M. A. and Zhang, Y. M. (2015). Linear model predictive control via feedback linearization for formation control of multiple wheeled mobile robots. In *IEEE International Conference on Information and Automation (ICIA)*, pages 1283–1288.

Kamel, M. A., Ghamry, K. A., and Zhang, Y. M. (2015). Fault tolerant cooperative control of multiple UAVs-UGVs under actuator faults. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 644–649.

Kamel, M. A. and Zhang, Y. M. (2015). Decentralized leader-follower formation control with obstacle avoidance of multiple unicycle mobile robots. In *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 406–411 **(Awarded the Best Conference Paper Award)**.

Kamel, M. A. and Zhang, Y. M. (2014). Developments and challenges in wheeled mobile robot control. In *International Conference for Intelligent Unmanned Systems (ICIUS)*.

Bibliography

Abdolhosseini, M., Zhang, Y. M., and Rabbath, C. A. (2013). An efficient model predictive control scheme for an unmanned quadrotor helicopter. *Journal of Intelligent & Robotic Systems*, 70(1):27–38.

Acevedo, J. J., Arrue, B. C., Maza, I., and Ollero, A. (2014). A decentralized algorithm for area surveillance missions using a team of aerial robots with different sensing capabilities. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4735–4740.

Antonelli, G., Arrichiello, F., and Chiaverini, S. (2009). Experiments of formation control with multirobot systems using the null-space-based behavioral control. *IEEE Transactions on Control Systems Technology*, 17(5):1173–1182.

Arora, J. S. (2012). *Introduction to Optimum Design*. Academic Press, Boston, third edition.

Balch, T. and Arkin, R. C. (1998). Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–9–39.

Beard, R. W., Lawton, J., and Hadaegh, F. Y. (2001). A coordination architecture for spacecraft formation control. *IEEE Transactions on Control Systems Technology*, 9(6):777–790.

Blažič, S. (2011). A novel trajectory-tracking control law for wheeled mobile robots. *Robotics and Autonomous Systems*, 59(11):1001–1007.

Blazic, S. (2014). On periodic control laws for mobile robots. *IEEE Transactions on Industrial Electronics*, 61(7):3660–3670.

Bloch, A. M. (2003). *Nonholonomic mechanics and control*, volume 24. Springer Science & Business Media.

Brockett, R. W. (1983). Asymptotic stability and feedback stabilization. In Brockett, R. W., Millman, R. S., and Sussman, H. J., editors, *Differential Geometric Control Theory*, page 181. Boston.

Camacho, E. F. and Bordons, C. (2007). *Model predictive control*. Springer London, London.

Campion, G., d'Andrea Novel, B., and Bastin, G. (1991). Controllability and state feedback stabilizability of non holonomic mechanical systems. In Canudas de Wit, C., editor, *Advanced Robot Control*, pages 106–124. Springer Berlin Heidelberg.

Chamseddine, A., Amoozgar, M., and Zhang, Y. M. (2014). Experimental validation of fault detection and diagnosis for unmanned aerial vehicles. In Valavanis, K. P. and Vachtsevanos, G. J., editors, *Handbook of Unmanned Aerial Vehicles*, pages 1123–1155. Springer Netherlands.

Chamseddine, A., Zhang, Y. M., and Rabbath, C. A. (2012). Trajectory planning and replanning for fault tolerant formation flight control of quadrotor unmanned aerial vehicles. In *American Control Conference (ACC)*, pages 3291–3296.

Chen, J., Sun, D., Yang, J., and Chen, H. (2010). A leader-follower formation control of multiple nonholonomic mobile robots incorporating receding- horizon scheme. *The International Journal of Robotics Research*, 29(6):727–747.

Chwa, D. (2010). Tracking control of differential-drive wheeled mobile robots using a backstepping-like feedback linearization. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 40(6):1285–1295.

Consolini, L., Morbidi, F., Prattichizzo, D., and Tosques, M. (2008). Leader-follower formation control of nonholonomic mobile robots with input constraints. *Automatica*, 44:1343–1349.

d'Andréa Novel, B., Campion, G., and Bastin, G. (1995). Control of nonholonomic wheeled mobile robots by state feedback linearization. *The International Journal of Robotics Research*, 14(6):543–559.

Das, T. and Kar, I. N. (2006). Design and implementation of an adaptive fuzzy logic-based controller for wheeled mobile robots. *IEEE Transactions on Control Systems Technology*, 14(3):501–510.

De La Cruz, C. and Carelli, R. (2006). Dynamic modeling and centralized formation control of mobile robots. In *Annual Conference of IEEE Industrial Electronics (IECON)*, pages 3880–3885.

De Luca, A. and Benedetto, M. D. D. (1993). Control of nonholonomic systems via dynamic compensation. *Kybernetika*, 29(6):593–608.

De Wit, C. C., Khennouf, H., Samson, C., and Sordalen, O. J. (1993). Nonlinear control design for mobile robots. In Zheng, Y.-F., editor, *Recent Trends in Mobile Robots*, volume 11, chapter 5, pages 121–156. Singapore: World Scientific.

Desai, J. P., Kumar, V., and Ostrowski, J. P. (1999). Control of changes in formation for a team of mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1556–1561.

Desai, J. P., Ostrowski, J., and Kumar, V. (1998). Controlling formations of multiple mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 2864–2869.

Dixon, W. E., Walker, I. D., and Dawson, D. M. (2001). Fault detection for wheeled mobile robots with parametric uncertainty. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, volume 2, pages 1245–1250.

Do, K. D. (2008). Formation tracking control of unicycle-type mobile robots with limited sensing ranges. *IEEE Transactions on Control Systems Technology*, 16(3):527–538.

Dong, W. and Guo, Y. (2007). Formation control of nonholonomic mobile robots using graph theoretical methods. In Grundel, D., Murphey, R., Pardalos, P., and Prokopyev, O., editors, *Cooperative Systems Control and Optimization*, pages 369–386. Springer Berlin Heidelberg.

Duan, H., Luo, Q., Shi, Y., and Ma, G. (2013). Hybrid particle swarm optimization and genetic algorithm for multi-UAV formation reconfiguration. *IEEE Computational Intelligence Magazine*, 8(3):16–27.

Duan, Z., Cai, Z., and Yu, J. (2005). Fault diagnosis and fault tolerant control for wheeled mobile robots under unknown environments: A survey. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3428–3433.

Duan, Z., Cai, Z., and Yu, J. (2006). Adaptive particle filter for unknown fault detection of wheeled mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1312–1315.

Ducard, G. J. J. (2007). *Fault-tolerant flight control and guidance systems for a small unmanned aerial vehicle*. PhD thesis, Eidgenössische Technische Hochschule (ETH Zürich).

Dunbabin, M. and Marques, L. (2012). Robots for environmental monitoring: Significant advancements and applications. *IEEE Robotics & Automation Magazine*, 19(1):24–39.

Dunbar, W. B. and Murray, R. M. (2006). Distributed receding horizon control for multi-vehicle formation stabilization. *Automatica*, 42(4):549–558.

Eberhart, R. C. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Sixth international symposium on micro machine and human science*, pages 39–43.

Egerstedt, M. and Hu, X. (2001). Formation constrained multi-agent control. *IEEE Transactions on Robotics and Automation*, 17(6):947–951.

Fax, J. A. and Murray, R. M. (2004). Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465–1476.

Ferreira, A., Pereira, F. G., Bastos-Filho, T. F., Sarcinelli-Filho, M., and Carelli, R. (2006). Avoiding obstacles in mobile robot navigation: Implementing the tangential escape approach. In *IEEE International Symposium on Industrial Electronics*, volume 4, pages 2732–2737.

Fierro, R. and Lewis, F. L. (1995). Control of a nonholonomic mobile robot: backstepping kinematics into dynamics. In *IEEE Conference on Decision and Control (CDC)*, volume 4, pages 3805–3810.

Furukawa, T., Durrant-Whyte, H. F., Bourgault, F., and Dissanayake, G. (2003). Time-optimal coordinated control of the relative formation of multiple vehicles. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 259–264.

Garrido, S., Moreno, L., and Lima, P. U. (2011). Robot formation motion planning using fast marching. *Robotics and Autonomous Systems*, 59(9):675–683.

Ge, S. S. and Fua, C.-H. (2005). Queues and artificial potential trenches for multirobot formations. *IEEE Transactions on Robotics*, 21(4):646–656.

Ghommam, J., Mehrjerdi, H., Saad, M., and Mnif, F. (2010). Formation path following control of unicycle-type mobile robots. *Robotics and Autonomous Systems*, 58(5).

Gu, D. and Hu, H. (2004). Model predictive control for simultaneous robot tracking and regulation. In *International Conference on Intelligent Mechatronics and Automation*, pages 212–217.

Hafez, A. T. (2014). *Design and implementation of modern control algorithms for unmanned aerial vehicles*. PhD thesis, Queen's University.

Hafez, A. T., Iskandarani, M., Givigi, S. N., Yousefi, S., Rabbath, C. A., and Beaulieu, A. (2014). Using linear model predictive control via feedback linearization for dynamic encirclement. In *American Control Conference (ACC), 2014*, pages 3868–3873.

Hu, J., Xu, J., and Xie, L. (2013). Cooperative search and exploration in robotic networks. *Unmanned Systems*, 1(01):121–142.

Isermann, R. (2006). *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer Berlin Heidelberg, New York.

Izadi, H. A., Gordon, B. W., and Zhang, Y. M. (2013). Hierarchical decentralized receding horizon control of multiple vehicles with communication failures. *IEEE Transactions on Aerospace and Electronic Systems*, 49(2):744–759.

Jiang, X., Motai, Y., and Zhu, X. (2005). Predictive fuzzy control for a mobile robot with nonholonomic constraints. In *International Conference on Advanced Robotics (ICAR)*, pages 58–63.

Jin, Z. and Murray, R. M. (2004). Double-graph control strategy of multi-vehicle formations. In *IEEE Conference on Decision and Control (CDC)*, volume 2, pages 1988–1994.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45.

Kamel, M. A. (2009). Drive automation interface of M113 armored personnel carrier. Master's thesis, Military Technical College.

Kanayama, Y., Kimura, Y., Miyazaki, F., and Noguchi, T. (1990). A stable tracking control method for an autonomous mobile robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 384–389.

Kanjanawanishkul, K., Hofmeister, M., and Zell, A. (2009). Smooth reference tracking of a mobile robot using nonlinear model predictive control. In *European Conference on Mobile Robots*, pages 161–166.

Keller, J. Y. and Darouach, M. (1997). Optimal two-stage Kalman filter in the presence of random bias. *Automatica*, 33(9):1745–1748.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98.

Kingston, D., Beard, R. W., and Holt, R. S. (2008). Decentralized perimeter surveillance using a team of uavs. *IEEE Transactions on Robotics*, 24(6):1394–1404.

Klančar, G., Matko, D., and Blažič, S. (2011). A control strategy for platoons of differential drive wheeled mobile robot. *Robotics and Autonomous Systems*, 59(2):57–64.

Klančar, G. and Škrjanc, I. (2007). Tracking-error model-based predictive control for mobile robots in real time. *Robotics and Autonomous Systems*, 55(6):460–469.

Koubaa, Y., Boukattaya, M., and Damak, T. (2013). Robust control of wheeled mobile robot in presence of disturbances and uncertainties. In *International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pages 274–280.

Kühne, F., Gomes, J., and Fetter, W. (2005). Mobile robot trajectory tracking using model predictive control. In *IEEE latin-american robotics symposium*.

Kuhne, F., Lages, W. F., and da Silva Jr, J. M. G. (2004). Model predictive control of a mobile robot using linearization. In *Proceedings of Mechatronics and Robotics*, pages 525–530.

Kumar, M., Cohen, K., and HomChaudhuri, B. (2011). Cooperative control of multiple uninhabited aerial vehicles for monitoring and fighting wildfires. *Journal of Aerospace Computing, Information, and Communication*, 8(1):1–16.

Lages, W. F. and Alves, J. A. V. (2006). Real-time control of a mobile robot using linearized model predictive control. In *IFAC Symposium on Mechatronic Systems*, pages 968–973.

Lawton, J. R. and Beard, R. W. (2002). Synchronized multiple spacecraft rotations. *Automatica*, 38(8):1359–1364.

Lawton, J. R. T., Beard, R. W., and Young, B. J. (2003). A decentralized approach to formation maneuvers. *IEEE Transactions on Robotics and Automation*, 19(6):933–941.

Lazar, M. (2006). *Model predictive control of hybrid systems: Stability and robustness*. PhD thesis, Technische Universiteit Eindhoven.

Lewis, M. A. and Tan, K.-H. (1997). High precision formation control of mobile robots using virtual structures. *Autonomous Robots*, 4(4):387–403.

Li, X., Xiao, J., and Tan, J. (2004). Modeling and controller design for multiple mobile robots formation control. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 838–843.

Lie, F. A. P. and Go, T. H. (2011). Reconfiguration control with collision avoidance framework for unmanned aerial vehicles in three-dimensional space. *Journal of Aerospace Engineering*, 26(3):637–645.

Lim, H., Kang, Y., Kim, C., Kim, J., and You, B.-J. (2008). Nonlinear model predictive controller design with obstacle avoidance for a mobile robot. In *IEEE/ASME International Conference on Mechtronic and Embedded Systems and Applications (MESA)*, pages 494–499.

Liu, Y. and Jia, Y. (2012). An iterative learning approach to formation control of multi-agent systems. *Systems & Control Letters*, 61(1):148–154.

Lofberg, J. (2000). A stabilizing mpc algorithm using performance bounds from saturated linear feedback. In *IEEE Conference on Decision and Control (CDC)*, volume 1, pages 644–649.

Luca, A. D., Oriolo, G., and Vendittelli, M. (2001). *Control of wheeled mobile robots: An experimental overview*, pages 181–226. Springer Berlin Heidelberg.

Ma, M.-M., Li, S., and Liu, X.-J. (2012). Tracking control and stabilization of wheeled mobile robots by nonlinear model predictive control. In *Chinese Control Conference (CCC)*, pages 4056–4061.

Mahony, R., Kumar, V., and Corke, P. (2012). Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics & Automation Magazine*, 19(3):20–32.

Marques, L., Martins, A., and de Almeida, A. T. (2005). Environmental monitoring with mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS)*, pages 3624–3629.

Martins, F. N., Celeste, W. C., Carelli, R., Sarcinelli-Filho, M., and Bastos-Filho, T. F. (2008). An adaptive dynamic controller for autonomous mobile robot trajectory tracking. *Control Engineering Practice*, 16(11):1354–1363.

Masoud, A. A. (2007). Decentralized self-organizing potential field-based control for individually motivated mobile agents in a cluttered environment: A vector-harmonic potential field approach. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 37(3):372–390.

Maurovic, I., Baotic, M., and Petrovic, I. (2011). Explicit model predictive control for trajectory tracking with mobile robots. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 712–717.

Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814.

Mead, R., Long, R., and Weinberg, J. B. (2009). Fault-tolerant formations of mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, pages 4805–4810.

Mehrjerdi, H., Saad, M., and Ghommam, J. (2011). Hierarchical fuzzy cooperative control and path following for a team of mobile robots. *IEEE/ASME Transactions on Mechatronics*, 16(5):907–917.

Mohammed, M. A. I. (2013). *Predictive Control of Electric Motors Drives for Unmanned Off-road Wheeled Vehicles*. PhD thesis, University of Ottawa.

Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., and Huhnke, B. (2008). Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597.

Moravec, H. P. (1980). Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, DTIC Document.

NaturalPoint, Inc. (2016). Flex-3 technical specifications. `http://www.optitrack.com/products/flex-3/specs.html`. Retrieved: August 14, 2016.

NAV Laboratory (2016). NAV Laboratroy YouTube Channel. `https://www.youtube.com/user/NAVConcordia/videos`. Retrieved: August 14, 2016.

Nieto-Granda, C., Rogers, J. G., and Christensen, H. I. (2014). Coordination strategies for multi-robot exploration and mapping. *The International Journal of Robotics Research*, 33(4):519–533.

Nijmeijer, H. and der Schaft, A. V. (1990). *Nonlinear dynamical control systems*. Springer.

Nilsson, N. J. (1969). A mobile automaton: An application of artificial intelligence techniques. Technical report, DTIC Document.

Ogata, K. (2010). *Modern control engineering*. Prentice Hall.

Ogren, P., Fiorelli, E., and Leonard, N. E. (2004). Cooperative control of mobile sensor networks:adaptive gradient climbing in a distributed environment. *IEEE Transactions on Automatic Control*, 49(8):1292–1302.

Oriolo, G., Luca, A. D., and Vendittelli, M. (2002). WMR control via dynamic feedback linearization: design, implementation, and experimental validation. *IEEE Transactions on Control Systems Technology*, 10(6):835–852.

Oya, M., Su, C.-Y., and Katoh, R. (2003). Robust adaptive motion/force tracking control of uncertain nonholonomic mechanical systems. *IEEE Transactions on Robotics and Automation*, 19(1):175–181.

Pan, Y. and Wang, J. (2012). Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. *IEEE Transactions on Industrial Electronics*, 59(8):3089–3101.

Panagou, D. and Kyriakopoulos, K. J. (2013). Cooperative formation control of underactuated marine vehicles for target surveillance under sensing and communication constraints. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1871–1876.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Englewood Cliffs, NJ.

Prasad, A., Sharma, B., and Vanualailai, J. (2015). A new stabilizing solution for motion planning and control of multiple robots. *Robotica*, FirstView:1–19.

Quanser, Inc. (2012). Quanser QGV User Manual. Technical report.

Rampinelli, V. T. L., Brandao, A. S., Sarcinelli-Filho, M., Martinsy, F. N., and Carelliz, R. (2010). Embedding obstacle avoidance in the control of a flexible multi-robot formation. In *IEEE International Symposium on Industrial Electronics (ISIE)*, pages 1846–1851.

Rao, S. S. (2009). *Engineering Optimization: Theory and Practice*. John Wiley & Sons, New Jersey, fourth edition.

Ren, W. and Beard, R. (2004). Decentralized scheme for spacecraft formation flying via the virtual structure approach. *Journal of Guidance, Control, and Dynamics*, 27(1):73–82.

Rodríguez-Seda, E. J., Tang, C., Spong, M. W., and Stipanović, D. M. (2014). Trajectory tracking with collision avoidance for nonholonomic vehicles with acceleration constraints and limited sensing. *The International Journal of Robotics Research*, 33(12):1569–1592.

Roumeliotis, S. I., Sukhatme, G., and Bekey, G. A. (1998). Fault detection and identification in a mobile robot using multiple-model estimation. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2223–2228.

Saber, R. O. and Murray, R. M. (2003). Flocking with obstacle avoidance: cooperation with limited communication in mobile networks. In *IEEE Conference on Decision and Control (CDC)*, volume 2, pages 2022–2028.

Samson, C. (1993). Time-varying feedback stabilization of car-like wheeled mobile robots. *The International journal of robotics research*, 12(1):55–64.

Sanfeliu, A., Hagita, N., and Saffiotti, A. (2008). Network robot systems. *Robotics and Autonomous Systems*, 56(10):793–797.

Saska, M., Vonásek, V., Krajník, T., and Přeučil, L. (2014). Coordination and navigation of heterogeneous mav-ugv formations localized by a hawk-eye-like approach under a model predictive control scheme. *The International Journal of Robotics Research*, 33(10):1393–1412.

Schwartz, I. (2000). Primus: autonomous driving robot for military applications. In *AeroSense 2000*, pages 313–323. International Society for Optics and Photonics.

Seyr, M. and Jakubek, S. (2005). Mobile robot predictive trajectory tracking. In *International Conference on Informatics in Control (ICINCO)*, pages 112–119.

Shojaei, K., Shahri, A. M., and Tabibian, B. (2013). Design and implementation of an inverse dynamics controller for uncertain nonholonomic robotic systems. *Journal of Intelligent & Robotic Systems*, 71(1):65–83.

Shojaei, K., Tarakameh, A., and Shahri, A. M. (2009). Adaptive trajectory tracking of wmrs based on feedback linearization technique. In *International Conference on Mechatronics and Automation (ICMA)*, pages 729–734.

Sira-Ramírez, H. and Castro-Linares, R. (2010). Trajectory tracking for non-holonomic cars: A linear approach to controlled leader-follower formation. In *49th IEEE Conference on Decision and Control (CDC)*, pages 546–551.

Sugihara, K. and Suzuki, I. (1996). Distributed algorithms for formation of geometric patterns with many mobile robots. *Journal of Robotic Systems*, 13(3):127–139.

Taheri-Kalani, J. and Khosrowjerdi, M. J. (2014). Adaptive trajectory tracking control of wheeled mobile robots with disturbance observer. *International Journal of Adaptive Control and Signal Processing*, 28(1):14–27.

Takahashi, H., Nishi, H., and Ohnishi, K. (2004). Autonomous decentralized control for formation of multiple mobile robots considering ability of robot. *IEEE Transactions on Industrial Electronics*, 51(6):1272–1279.

Tanner, H. G., Loizou, S. G., and Kyriakopoulos, K. J. (2003). Nonholonomic navigation and control of cooperating mobile manipulators. *IEEE Transactions on Robotics and Automation*, 19(1):53–64.

Tanner, H. G., Pappas, G. J., and Kumar, V. (2004). Leader-to-formation stability. *IEEE Transactions on Robotics and Automation*, 20(3):443–455.

Tao, P. and Shan, L. (2014). Formation control of multiple wheeled mobile robots via leader-follower approach. In *Chinese Control and Decision Conference (CCDC)*, pages 4215–4220.

Thorpe, C., Hebert, M. H., Kanade, T., and Shafer, S. A. (1988). Vision and navigation for the carnegie-mellon navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362–373.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., and Hoffmann, G. (2006). Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692.

Thumati, B. T., Dierks, T., and Sarangapani, J. (2012). A model-based fault tolerant control design for nonholonomic mobile robots in formation. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 9(1):17–31.

Tzafestas, S. G. (2014). *Introduction to Mobile Robot Control*. Elsevier.

van Essen, H. A. and Nijmeijer, H. (2001). Non-linear model predictive control for constrained mobile robots. In *European Control Conference (ECC)*, pages 1157–1162.

Wang, J. and Xin, M. (2013). Integrated optimal formation control of multiple unmanned aerial vehicles. *IEEE Transactions on Control Systems Technology*, 21(5):1731–1744.

Wang, Z., He, Y., and Han, J. (2011). Distributed receding horizon formation control for multi-vehicle systems with relative model. In *IFAC World Congress*, pages 13576–13581.

Xie, F. (2007). *Model predictive control of nonholonomic mobile robots*. PhD thesis, Oklahoma State University.

Xie, F. and Fierro, R. (2008). First-state contractive model predictive control of nonholonomic mobile robots. In *American Control Conference (ACC)*, pages 3494–3499.

Xiong, W., Chen, Z., and Zhou, R. (2007). Optimization for multiple flight vehicles formation reconfiguration using hybrid genetic algorithm. In *Chinese Guidance, Navigation and Control Conference*, pages 501–506.

Xu, Q., Yang, H., Jiang, B., Zhou, D., and Zhang, Y. M. (2014). Fault tolerant formations control of UAVs subject to permanent and intermittent faults. *Journal of Intelligent & Robotic Systems*, 73(1-4):589–602.

Yang, H., Jiang, B., Yang, H., and Zhang, K. (2015). Cooperative control reconfiguration in multiple quadrotor systems with actuator faults. In *IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS)*, pages 386–391.

Yang, H., Jiang, B., and Zhang, Y. M. (2014). Fault-tolerant shortest connection topology design for formation control. *International Journal of Control, Automation and Systems*, 12(1):29–36.

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

Yu, X. and Jiang, J. (2015). A survey of fault-tolerant controllers based on safety-related issues. *Annual Reviews in Control*, 39(1):46–57.

Yu, X., Liu, Z., and Zhang, Y. M. (2016). Fault-tolerant formation control of multiple UAVs in the presence of actuator faults. *International Journal of Robust and Nonlinear Control*, 26(12):2668–2685.

Yuan, C., Zhang, Y. M., and Liu, Z. (2015). A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques. *Canadian Journal of Forest Research*, 45(7):783–792.

Yun, X. and Yamamoto, Y. (1992). On feedback linearization of mobile robots. *Technical Reports (CIS)*, page 503.

Zhang, Y. M. and Jiang, J. (2002). Active fault-tolerant control system against partial actuator failures. *IEE Proceedings-Control Theory and Applications*, 149(1):95–104.

Zhang, Y. M. and Jiang, J. (2003). Fault tolerant control system design with explicit consideration of performance degradation. *IEEE Transactions on Aerospace and Electronic Systems*, 39(3):838–848.

Zhang, Y. M. and Jiang, J. (2008). Bibliographical review on reconfigurable fault-tolerant control systems. *Annual Reviews in Control*, 32(2):229–252.

Zhang, Y. M. and Mehrjerdi, H. (2013). A survey on multiple unmanned vehicles formation control and coordination: Normal and fault situations. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1087–1096.