Initialisation and Decentralised Control for Robotic Formation

Anh Duy Nguyen

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

ARC Centre of Excellence for Autonomous Systems

Faculty of Engineering

University of Technology, Sydney, Australia

June 2006

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Candidate

Production Note:
Signature removed prior to publication.

Anh Duy Nguyen

To my mother and my father

To my wife and my son

Abstract

This thesis addresses the problem of coordination of a group of mobile robots to get into and maintain a formation with a desired shape. Two issues of particular importance in the coordination problem are formation initialisation and maintenance. Inter-robot collision and communication bandwidth limitations raise certain difficulties and require a thorough treatment. This thesis presents original contributions towards a solution to the formation initialisation and maintenance for multiple mobile robots in an obstacle-free environment.

In the context of robotic formation control, the commonly-used virtual robot tracking combined with *l-l* control has limitations in the establishment of a line formation, the possibility of collision between robots, and the singularity cases involved. A new approach called the *Virtual Head Robot Tracking* (VHRT) and *Three Point l-l* (3PLL) control incorporated with reactive control schemes is presented. The approach represents an appropriate solution to formation control for a group of three mobile robots with singularities alleviated and inter-robot collision completely avoided.

For a group of more than three robots, a step-by-step procedure is proposed allowing the robots in turn to participate in the process of formation initialisation, that is based on a predefined control graph, while ensuring inter-robot collision avoidance.

An observed-based decentralised control approach is proposed to establish and maintain a desired formation in the condition of a limited information exchange among robots in the group. This suggests the capability of enlarging the size of the platoon of vehicles in practice.

The theoretical work of the thesis is evaluated by extensive simulations of multiple mobile robots based on their kinematic models. The results obtained are also experimentally tested, in part, on a group of two Amigo mobile robots.

List of symbols

Symbol	Description	Unit
b	The half distance between two wheels of a robot	m
$-b_h, b_h$	Hardware bound on the steering angle in car-like vehicle model	rad
$Card(S,N)$	The number of elements of a set S of control graphs of N robots	-
D	Distance between the castor and the centre of axis of the wheels of a robot	m
$D_{\max}$	Largest distance from one of three head points of a robot to its centre	m
d	Distance between two centre points of head robot and its host	m
e_x	Horizontal coordinate error between a virtual robot of robot j and robot i in *virtual robot tracking* control	m
$e_x(0)$	Initial horizontal coordinate error between a virtual robot of robot j and robot i in *virtual robot tracking* control	m
e_y	Vertical coordinate error between a virtual robot of robot j and robot i in *virtual robot tracking* control	m
$e_y(0)$	Initial vertical coordinate error between a virtual robot of robot j and robot i in *virtual robot tracking* control	m
e_{xji}	Horizontal coordinate error between a head robot of robot j and a virtual robot of robot i in *virtual head robot tracking* control	m
$e_{xji}(0)$	Initial horizontal coordinate error between a head robot of robot j and a virtual robot of robot i in *virtual head robot tracking* control	m
e_{yji}	Vertical coordinate error between a head robot of robot j and a virtual robot of robot i in *virtual head robot tracking* control	m

Symbol	Description	Unit
$e_{yji}(0)$	Initial vertical coordinate error between a head robot of robot j and a virtual robot of robot i in *virtual head robot tracking* control	m
F	Input force of a robot	N
h	Platoon level function	-
i, j	Index of a robot	-
I_{r_i}	An identity matrix of dimension r_i.	-
J	Moment of inertia of a robot.	$kg.m^2$
k, l	Index of a step in initialisation process	-
l	Longitudinal clearance from a virtual robot to its host in *virtual robot tracking* control	m
L	Longitudinal clearance from a virtual robot to its host in *virtual head robot tracking* control	m
l_K	Longitudinal clearance from point K to the centre point of a robot	m
l_{ij}	Distance between the centre of robot i and the reference point of the robot j	m
$l_{ij}(0)$	Initial distance between the centre of robot i and the reference point of the robot j	m
l_{ij}^d	Desired length between robot i and robot j	m
l_{ijK}^d	Desired length between robot i and robot j with respect to virtual point K	m
m	Mass of a robot	kg
m_i	The number of input signals of ith control agent	-
m_Σ	The number of input signals of a platoon	-
N	The number of robots in a group	-
n_k	The number of steps of the formation initialisation process	-
n_Σ	The number of state variables of a platoon	-
p_i	The observer order for ith control agent	-

Symbol	Description	Unit
p, q	Control parameter – positive constant	-
r	Clearance along rear wheel axis from a virtual robot to its host in *virtual robot tracking* control	m
R	Clearance along rear wheel axis from a virtual robot to its host in *virtual head robot tracking* control	m
r_i	The number of output signals of ith control agent	-
r_Σ	The number of output signals of a platoon	-
r_K	Clearance along rear wheel axis from point K to the centre point of a robot	m
r_{safe}	Radius of a virtual safe circle covering a robot	m
r_w	Radius of the wheel of a robot	m
$S_1(N)$	The set of allowable control graphs of N robots in which at the final step there is one robot to become active	-
$S_2(N)$	The set of allowable control graphs of N robots in which at the final step there is two robots to become active	-
T_r	Finite reaching time for *l-l* control using terminal attractor	s
u_1, u_2	Input signal of a robot	-
u_i	The velocity vector of robot i	-
$u_i(t)$	The input vector of the i-th agent	-
v	Translational velocity of a robot	m/s
v_1, v_2	Horizontal and vertical velocity of a robot respectively	m/s
v_i	Translational velocity of robot i	m/s
v_{vi}	Translational velocity of a virtual robot of robot i	m/s
x	Horizontal coordinate of the centre point of a robot	m
x_i	Horizontal coordinate of the centre point of robot i	m
$x_i(0)$	Initial horizontal coordinate of the centre point of robot i	m
x_{ci}	Horizontal coordinate of the castor of robot i	m
x_{hj}	Horizontal coordinate of the centre point of a head robot of robot j	m

Symbol	Description	Unit
x_{vi}	Horizontal coordinate of the centre point of a virtual robot of robot i	m
y	Vertical coordinate of the centre point of a robot	m
y_i	Vertical coordinate of the centre point of robot i	m
$y_i(0)$	Initial vertical coordinate of the centre point of robot i	m
$y_i(t)$	The output vector of the i-th agent	-
y_{ci}	Vertical coordinate of the castor of robot i	m
y_{hj}	Vertical coordinate of the centre point of a head robot of robot j	m
y_{vi}	Vertical coordinate of the centre point of a virtual robot of robot i	m
w_{ij}	Weight of edge ij of a weighted digraph	-
α_1, α_2	Control parameter, positive constant	s^{-1}
α_1^*, α_2^*	Control parameter in modified *l-l* control	$m^{(1-p/q)}s^{-1}$
$\alpha_{1K}^*, \alpha_{2K}^*$	Control parameter in *three point l-l* control with respect to virtual point K	$m^{(1-p/q)}s^{-1}$
δ_1, δ_2	Control parameter for collision avoidance	m
Δx_{ij}	Horizontal coordinate error between the castor of robot j and the centre of robot i	m
Δy_{ij}	Vertical coordinate error between the castor of robot j and the centre of robot i	m
θ	Orientation of a robot	rad
θ_i	Orientation of robot i	rad
$\theta_i(0)$	Initial orientation of robot i	rad
$\theta_i(t)$	Orientation of robot i at time point t	rad
θ_{hj}	Orientation of a head robot of robot j	rad
θ_{vi}	Orientation of a virtual robot of robot i	rad
λ_1, λ_2	Control parameter – positive constant	s^{-1}

Symbol	Description	Unit
$\lambda_{j1}, \lambda_{j2}$	Control parameter for robot j – positive constant	s^{-1}
ρ	Distance between two robots	m
τ	Input torque of a robot	N.m
ϕ	Rolling angle of the wheel in Roller unicycle model	rad
ϕ_{st}	Steering angle in car-like vehicle model	rad
ψ_{ij}	Relative angle between robot i and robot j	rad
ψ_{ijK}	Relative angle between robot i and robot j with respect to virtual point K of robot j	rad
$\psi_{ij}(t)$	Relative angle between robot i and robot j at time point t	rad
$\psi_{ij}(0)$	Initial relative angle between robot i and robot j	rad
ψ_{ij}^{d}	Desired relative angle between robot i and robot j	rad
ω	Angular velocity of a robot	rad/s
ω_1, ω_2	Left and right wheel rotational velocity respectively	rad/s
ω_i	Angular velocity of robot i	rad/s
ω_{vi}	Angular velocity of a virtual robot of robot i	rad/s
$\Im_i(t)$	Information available to the ith control station	-

Abbreviations

3PLL	Three Point *l*-*l*
ARIA	Advanced Robotics Interface Application
AUV	Autonomous Unmanned Vehicle
DOF	Degree of Freedom
GA	Genetic Algorithm
HR	Head Robot
ID	Identification
LAS	Linear Autonomous System
LFS	Leader-to-Formation Stability
MDLe	extended Motion Description Language
MPC	Model Predictive Control
PC	Personal Computer
PTZ	Pan Tilt Zoom
RFP	Formation Reference Point
RL	Reinforcement Learning
SIP	Server Information Packet
TCP	Transmission Control Protocol
V-GPS	Vision-based Global Positioning System
VHRT	Virtual Head Robot Tracking
VR	Virtual Robot
VRT	Virtual Robot Tracking
VS	Virtual Structure
WMR	Wheeled Mobile Robot

Contents

List of figures

Chapter 1

Introduction

The research project reported in this thesis is devoted to the coordination of a group of mobile robots to get into and maintain a formation with a desired shape. This chapter presents an overview of the work covered in the project. An introduction to robotic formation, which provides motivation for this research, is given in the first section of this chapter. Section 1.2 then discusses issues relating to main problems in robotic formation and defines the problems to be addressed in the thesis. Section 1.3 presents robotic formation research categories. Thesis scope is described in Section 1.4. Section 1.5 summarises the principal contributions of the work. Finally, the chapter concludes with an overview of the thesis organisation.

1.1 Robotic formation

Formation may have an essential role for activities of groups in both natural and artificial world. In nature, formation behaviours, such as flocking and schooling, benefit a group of animals performing them in various ways. For example, each animal in a formation herd may minimise its encounters with predators by increasing the chance of detecting flock's enemies. Birds, such as Canada geese, sometimes fly in inverted V-formations, which provide them with aerodynamic advantages that enable them to fly long distances with less fatigue [Bekey, 2005]. It has now been showed that

when white pelicans are trained to fly in formation with proper spacing, their heart rates (and hence energy expenditures) drop by about 30% [Weimerskirch *et al.*, 2001]. Grouped animals also can combine their sensing capabilities to maximise the efficiency of foraging for food. Examples of pattern formation in animals include bird flocking, fish schooling, and ant forming chain [Camazine *et al.*, 2001].

Studies of flocking and schooling show that these behaviours emerge as a combination of a desire to stay in the group and to, simultaneously, keep a separation distance from other members of the group. Since groups of artificial agents could similarly benefit from formation tactics, robotic researchers and those in artificial life community have been inspired with these biological studies to develop formation behaviours for both simulated agents and robots.

Formation is also very important in many military applications where sensor assets are limited as making a formation allows each robot to concentrate its sensing capability on a portion of environment, while the other robots in the formation cover the rest. For instance, fighter pilots direct their visual and radar search responsibilities depending on their positions in a formation. Robotic scouts are also useful by directing their sensors in different areas to ensure full coverage. Approaches to robotic formation problem are of potential applications in many other fields, such as search and rescue operations, agricultural coverage tasks, landmine removal, remote terrain and space exploration, and also the control of satellites and unmanned aerial vehicles.

The concept of multiple autonomous unmanned vehicles-AUVs (land, air, or underwater) operating in formation is emerging as a key technology in mobile robotics that has been the focus of intense research effort over recent years. The formation control approach has several advantages over the traditional monolithic agent control, including overall system robustness, intelligence, enhanced performance (increased instrument resolution, reduced cost), and flexibility (reconfigurable capability, fault tolerance). Some typical applications include moving large objects [Donald *et al.*, 2000], exploration [Fox *et al.*, 2000], surveillance [Feddema and Schoenwald, 2002], search and rescue [Jennings *et al.*, 1997], and deep space observation [Wang and Hadaegh, 1996].

With the development of powerful control techniques for single vehicles, recent advances in sensors, energy storage devices, the explosion in computation and communication capabilities, and the advent of miniaturisation technologies, it is perceived that large groups of automated vehicles can now be coordinated in an effective manner for a variety of tasks which are beyond the ability of individual vehicles.

1.2 Robotic formation problems

Robotic formations are defined as groups of mobile robots establishing and maintaining some predetermined geometrical shape by controlling the positions and orientations of individual robots relative to the group, at the same time allowing the group to move as a whole. According to Lemay *et al.*(2004), that purpose requires to solve the following issues:

1) How to initialise and establish a formation? Robots assemble at the arbitrary starting points and/or establish the formation.

2) How to avoid inter-robot collision? When robots are used to perform tasks in a shared workspace, each one becomes a mobile obstacle for the other. Inter-robot collision avoidance consequently is an essential consideration for multi robot coordination.

3) How to maintain formation shape while moving? Criteria like the stability while moving in formation and robustness to robot failure are important.

4) How to change the shape of the formation? The group of mobile robots in a formation sometimes has to change its shape to avoid obstacles or to fulfil required tasks.

5) How to avoid obstacles? The group of mobile robots occasionally split/deform and then re-establish the formation or preserve the shape of the formation.

6) Is it possible to realise any types of formation? Some geometric conditions have to be considered to guarantee formation feasibility given the individual robot kinematics.

The above problems can be solved in simple cases, which involve spatial coordination among robots, or in the more complex ones, which add temporal coordination of robots' trajectories and roles.

In forming and maintaining the multi-robot pattern there is generally a trade-off between precision and feasibility on one side, and between the necessity of global information and communication capacity on the other side. Those systems that require global information or broadcast communication may have a lack of scalability or high costs of the physical set-up but allow for more accuracy in forming a large range of geometrical shapes [Carpin and Parker, 2002; Sugihara and Suzuki, 1996]. On the contrary, systems using only local communication and sensor data, while limited in variety and precision of formations, tend to be more scalable, more robust, and easier to build [Balch and Hybinette, 2000; Desai, 2001].

In the literature, there is not much research on the formation initialization issue in multi-robot coordination. A solution that addresses mainly first three above issues of robotic formation problems, which are how to establish and maintain a formation for a group of mobile robot without collisions among them, is the subject of the research reported in this thesis. Within the scope of the thesis, attention will be restricted to the coordination of mobile robots in formations in the obstacle-free environment.

1.3 Robotic formation research categories

According to Michaud *et al.*(2002), characteristics of the research in robotic formation can be grouped into three categories: perceptual, shape and control.

1.3.1 Perceptual characteristics

Visibility of other robots Approaches may consider *complete* or *limited* visibility of robots in the groups. This relates to two different classes of sensors have emerged for

mobile robots: visual sensors, which use light reflected from objects in the environment to reason about structure, and nonvisual sensors, which use various audio, inertial, and other modalities to sense the environment.

Frame of reference Robots can use an *absolute* positioning system or their own *relative* reference to decide and take their actions. An absolute positioning system uses fixed and known beacons. To do so, it is necessary to cover the whole robot world with a beacon system so that it can compute its position at any moment. Using a relative position system, a robot defines by itself its references which are either characteristic features of the environment or objects known with good accuracy.

Communication capabilities Either *no communication*, communication of *global information* or of *local information* is possible. In addition, distinctions between implicit and explicit communication are usually made, in which implicit communication occurs as a side effect of other actions, or "through the world", whereas explicit communication is a specific act designed solely to convey information to other robots on the team.

1.3.2 Shape characteristics

Types of formation This relates to the variety of geometrical shapes that an approach can handle. Some regular formation types are *circle*, *diamond*, *wedge*, *line*, *column*, *triangle*, *rectangle*, *arrowhead*, *hexagon*, *tree*, and *lattice* (hexagonal, rectangular or triangular) (see Figure 1.1).

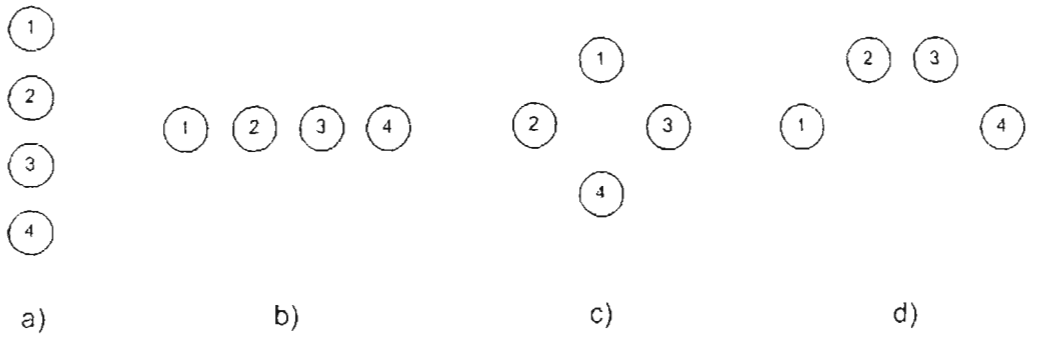


Figure 1.1: Some types of formations: *column, line, diamond, wedge.*

Position determination Three techniques exist: *unit-centre-referenced*, where the average of the x and y position of all robots is computed and used as a common reference; *point-referenced*, with each robot determining its position in relation to a single point, which can be the leading robot or a "virtual" point; and *neighbour-referenced*, in which each robot maintains a position relative to one or up two robots in proximity (see Figure 1.2).

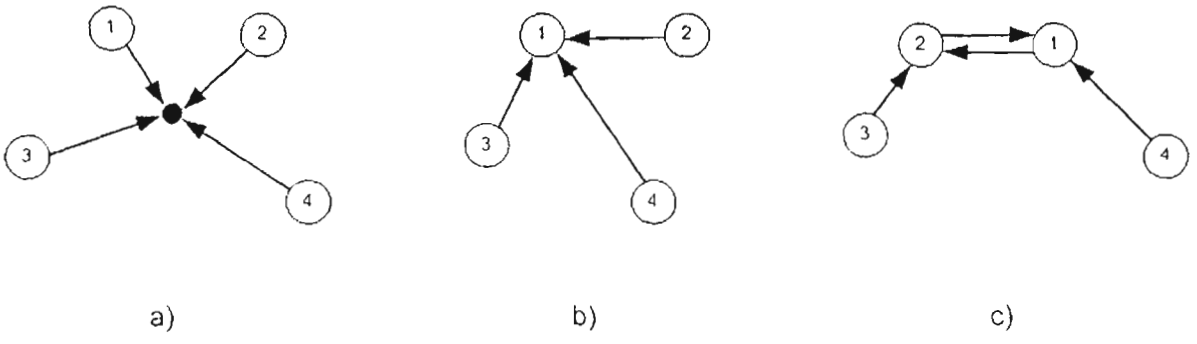


Figure 1.2: Techniques for position determination: *unit-centre, point, and neighbour-referenced*

Structural constraints Formation can be *rigid*, i.e., their shape must be preserved at all time, or *flexible*.

1.3.3 Control characteristics

Decision process Based on the control process involved, there are two groups. The first group includes approaches where the coordination is done by a *centralised* unit that supervises the whole group and command the individual robots. Studies in the second group use *distributed* methods for achieving the coordination. In those approaches, the decision process can be either *homogeneous* if all robots follow the same decision rules, or *heterogenous* if the robots have different reactions to make their decisions.

Dependence on temporal states The algorithm can be *oblivious* if its decisions are determined only from the sensor information obtained at the time instant, or *non-oblivious* if it exploits information from the past.

Control strategy It can be based on *feedback control laws* (involving for instance input-output feedback linearization, observer-based decentralised control, etc.), *robotic*

behaviours, or on *hybrid platforms* (i.e. involving a supervisory level and an execution level).

Another perspective to categorize formations is distinguishability of mobile robots [Hsu and Liu, 2005]. If robots in a formation can be distinguished from each other based on their inner states, generally by identification (ID), we call these robots ID-robots and call their formations ID-formations. In contrast, if robots cannot be distinguished, we call such robots anonymous robots (marked as ANO-robots) and the formations anonymous formation (marked as ANO-formation).

1.4 Thesis scope

The objectives of this research are to propose a new approach to initialise and maintain a desired shape of a group of mobile robots. The group of three mobile robots will be initialised to get into and maintain a desired formation using an algorithm, which combines *virtual head robot tracking* and *three point l-l* control incorporated with a reactive control scheme. This algorithm ensures that any desired formation can be achieved for a group of three mobile robots without inter-robot collision while satisfying the limitation of communication range. A step-by-step procedure will be developed to extend this algorithm for a group of N mobile robots. In addition, to accommodate the restriction in information exchange, a decentralized approach will also be proposed to implement the global feedback controller for the robots moving in a formation by using linear functional observers. This method seems to be an appropriate solution for maintaining a desired formation in the condition of limited information exchanged among robots in the group. The application of proposed approaches in this thesis is restricted for the coordination of mobile robots in formations in the obstacle-free environment.

Extensive simulations for a group of multiple mobile robots and laboratorial experiments on Amigo mobile robots are carried out to validate the proposed approaches.

1.5 Thesis contribution

The main contribution of this thesis is the development of decentralised controllers to initialise and maintain a desired formation of multiple mobile robots in an obstacle-free environment. The specific contributions are:

- A new *Virtual Head Robot Tracking* (VHRT) control to establish any desired configuration of two robots without collisions between them.
- *Three Point l-l* (3PLL) control to avoid collision among three mobile robots with the singularity, which is a characteristic of *l-l* control, entirely alleviated.
- A procedure to initialise a chosen robotic formation together with the use of control graphs for a group of N mobile robots.
- An observed-based decentralised approach to deal with the communication bandwidth limitations in the coordination control of multiple mobile robots.

There are some control laws used to establish a desired configuration of two mobile robots such as $l-\psi$ control [Desai *et al.*, 1998], *separation-bearing* control [Fierro *et al.*, 2001; Fierro *et al.*, 2002]. The *Virtual Robot Tracking* (VRT) control, proposed in [Jongusuk and Mita, 2001], could be used for that purpose with the consideration of inter-robot collision avoidance. However this approach has limitations in the establishment of a line configuration and the possibility of collisions between robots in some cases. The VHRT control is proposed to overcome these circumstances and obtain a desired leader-follower configuration of two mobile robots.

For a group of three mobile robots, a modified *l-l* control is used for collision avoidance among robots [Jongusuk and Mita, 2001]. This control law, as well as the original *l-l* control [Desai *et al.*, 1998] and *separation-separation* control [Fierro *et al.*, 2001; Fierro *et al.*, 2002], subject to a singularity when three robots lie on the same line connecting them. The 3PLL control is proposed to deal with this case. An algorithm using VHRT and 3PLL control incorporated with a reactive control scheme then can be

used to initialize and maintain a desired formation for a group of three mobile robots without inter-robot collisions.

Extending the proposed algorithm for three mobile robots, a step-by step procedure is proposed to initialise a formation for a group of N mobile robots. Under some assumptions, the procedure ensures that a desired formation of multiple mobile robots can be established from arbitrary robots' positions without collisions among them.

Finally, using linearised models of robots around specific trajectories, an observed-based decentralised approach is also proposed to establish and maintain a desired formation in the condition of limited information exchanged among robots in the group. This brings the capability of enlarging the size of the platoon of vehicles in practice.

Peer-reviewed publications resulted from this project include:

Conference papers

1. Ha, Q. P., Nguyen, A. D., and Trinh, H. "Simultaneous State and Input Estimation with Application to a Two-Link Robotic System" *Proc. The 5th Asian Control Conference*, Melbourne, Australia, pp. 322-328, 20-23 July, 2004.

2. Nguyen, A. D., Ha, Q. P., Huang, S., and Trinh, H. "Observer-Based Decentralized Approach to Robotic Formation Control" *Proc. Australasian Conference on Robotics and Automation*, Canberra, Australia, pp. 1-8, 6-8 December, 2004.

3. Ngo, V. T., Nguyen, A. D., and Ha, Q. P. "Integration of planning and control in robotic formations" *Proc. Australasian Conference on Robotics and Automation*, Sydney, Australia, pp. 1-8, 5–7 December, 2005.

4. Ha, H. M., Nguyen, A. D., and Q.P. Ha, "Controlling formation of multiple mobile robots with inter-robot collision avoidance" *Proc. Australasian Conference on Robotics and Automation*, Sydney, Australia, pp. 1-8, 5–7 December, 2005.

5. Ngo, V.T., Nguyen, A.D., and Ha, Q.P. "Toward a generic architecture for robotic formations: planning and control" *Proc. International Conference on Intelligent Technologies*, Phuket, Thailand, pp. 89-96, 14 -16 December, 2005 (Best Paper Award).

6. Nguyen, A. D., Ha, Q. P., and Nguyen, H. T. "Virtual Head Robot Tracking and Three-Point l-l Control for Multiple Mobile Robots" *Proc. IEEE Workshop on Distributed Intelligent Systems*, Prague, Czech Republic, pp. 73-78, 14-16 June, 2006.

7. Nguyen, A. D., Kwok, N. M., Ngo, V. T., and Ha, Q. P. "Collision-Free Formations with Reactively-Controlled Virtual Head Robot Tracking" *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, pp. 2509-2514, 9-15 October, 2006.

Journal papers

1. Nguyen, A. D., Ngo, V. T., Ha, Q. P., and Dissanayake, G. "Robotic Formation: Initialisation, Trajectory Planning, and Decentralised Control" *International Journal of Automation and Control* (IJAC), accepted February 2006.
2. A.D. Nguyen, V.T. Ngo, N. M. Kwok and Q.P. Ha, "Formation Initialisation using Virtual Head Robot Tracking and Three-Point *l-l* Control," *International Transactions on Systems Science and Applications* (ITSSA), submitted August 2006.

A citation on our work can be found in [Lee *et al.*, 2005].

1.6 Organisation of the thesis

A short description of the chapters following this introduction is given below.

Chapter 2: Robotic formation control: a short survey

This chapter reviews models of mobile robots widely used for control analysis and design and related work in the field of robotic formation. Through the review, existing techniques are examined with their advantages and disadvantages.

Chapter 3: Virtual head robot tracking control

A new *virtual head robot tracking* control to establish a desired configuration between two robots considering inter-robot collision avoidance is presented in this chapter. The $l-\psi$, *separation-bearing*, and *virtual robot tracking* control are described and analysed. A new approach with the capability of establishing a desired configuration without inter-robot collisions is validated by simulations with a group of two robots.

Chapter 4: Three point *l-l* control for formation control

This chapter proposes a new *three point l-l* control for collision avoidance in a group of three mobile robots. The original *l-l*, *separation-separation*, and modified *l-l* control with their singularity are described. An algorithm, which combines VHRT and 3PLL incorporated with a reactive control scheme, is also proposed to establish and maintain formations of a group of three mobile robots taking into account the requirement of collision avoidance among them. Simulations with a group of three mobile robots are used to demonstrate the validity of the proposed approach.

Chapter 5: Formation initialisation for a group of N mobile robots

Chapter 5 proposes a step-by-step procedure to establish formations for a group of mobile robots. Under some assumptions, the procedure ensures that a desired formation of multiple mobile robots can be obtained without inter-robot collisions. The graph concept is also used to model formations. Extensive simulations with five robots validate the capability of establishing formations for a group of more than three robots without collisions among them.

Chapter 6: Observer-based decentralized approach to robotic formation

To accommodate the restriction in information exchange, a new observer-based decentralised control approach to robotic formation is proposed in this chapter. This

method implements a global feedback controller for the robots in a formation using linear functional observers. The proposed approach is tested through simulations and experiments for the control of non-holonomic mobile robots.

Chapter 7: Thesis summary and conclusion

The final chapter of this thesis summarises the conclusions drawn from the research reported here, as well as advancing a number of recommendations for future work.

Appendices

In the appendices, specifications of the testbed- Amigo Mobile Robot- are given. A presentation of ARIA (Advanced Robotics Interface for Applications) C++ class, an algorithm for Matlab-Simulink simulation and an organisation of the accompanying CD-ROM for video clips are also included.

Chapter 2

Robotic formation control: a short survey

This chapter, reviewing the commonly-used models of mobile robots, summarises control approaches in the literature available for analysis and design of robotic formation.

2.1 Mobile robots and mathematical models

2.1.1 Mobile robots

Mobile robots simply refer to robots that can change their location through locomotion. There are a number of different types of mobile robots. On the basis of application domain, there are four main categories of mobile robots [Dudek and Jenkin, 2000]:

- **Terrestrial**: Robots that run or walk on the ground. They are designed to operate on solid surface and take advantage of gravity. Terrestrial mobile robots include wheeled, tracked, legged and some other kinds. Terrestrial robots are also known as ground–contact robots.

- **Aquatic**: Robots that operate in water or at water surface. Most aquatic robots use water jets or propellers to move around. Aquatic robots are important in various applications, especially in the ocean, as there are many resources in the ocean waiting to be exploited.

- **Airborne**: Robots that mimic existing aircraft or birds, for instance, robotic helicopters, controlled parachutes, fixed-wing aircraft. The issue of maintaining enough energy to remain stationary is of importance with flying robots.

- **Spatial**: Space robots, as their name suggests, are designed to operate in outer space. Two main classes of space robots are climbing robots and those that can self-propel.

One issue that sets mobile robotics aside from other research areas such as manipulator robotics, artificial intelligence or computer vision is the problem associated with the understanding of large-scale space, or the working region that is larger than what can be observed from any certain point. To deal with this problem implies dealing with incremental acquisition of information, estimation of positional error, real-time response. It is important that these functionalities are synchronized in a global manner, in order to perform fundamental tasks of mobile robots, including moving through, sensing about and reasoning about the working environment.

Mobile robots are well suited for tasks that exhibit one or more of the following characteristics:

- Hostility: a task involving an inhospitable environment for human beings.

- Inaccessibility: a task involving a remote environment into which sending a human operator would be too difficult or would take too long or an environment inaccessible to humans such as microscopic environments.

- Human unsuitability: a task with high duty cycle or high fatigue factor, or a task that is highly disagreeable to a human.

Based on their locomotion mechanics, mobile robots can be classified into two main groups: legged and wheeled [Siegwart and Nourbakhsh, 2004]. The wheel has been by far the most popular locomotion mechanism in mobile robotics and in man-made vehicles in general. Wheeled robots are mechanically simple, easy to construct, and have a favourable weight-to-mechanism ratio. In addition, they are simpler to control, pose fewer stability problems, use less energy per unit distance of motion, and can travel with significant speed. Wheeled mobile robots, as in the majority of robotic formation research, are concerned in this thesis.

2.1.2 Mathematical models of mobile robots

Mathematic models lie at the foundation of any science: physics, finance, electronics, chemistry, and so on. They allow us to make an abstraction of a real world system and use this abstraction to understand the system. The benefits of a model depend on two factors: how well the model predicts the real world and how simple the model is. The simpler, or less complicated, the better, and the more accurate, the better.

As some properties can be proven only in the context of a mathematical model, i.e. given a set of assumptions some specific set of conclusions can be drawn, the applicability of these conclusions to the real world is then depending on the accuracy of the used mathematical model and the validity of the assumptions.

Some of the most common mathematical models of mobile robots will be presented in the rest of this section.

Kinematic (first order) model

This simplest model describes a point-like robot moving around in the plane:

$$\begin{aligned} \dot{x} &= u_1, \\ \dot{y} &= u_2, \end{aligned} \tag{2.1}$$

or

$$\dot{z} = u,$$

where $z,u \in R^2$, $z=(x,y)^T$ is the position and $u=(u_1,u_2)^T$ is the input of the robot. This model is called kinematic or first order as there is a maximum of one integration from input to state. In many real configurations, the velocity is limited with an input bound $\|u\| \le u_{\max}$.

This model is simple but the main drawback is that it allows instantaneous velocity changes; and hence it does not accurately describe inertial vehicles, particularly those with a low power to weight ratio.

Dynamic (second order) model

The obvious fix of the "instant velocity changes" problem is to incorporate inertia into the model. This corresponds to

$$\begin{aligned} \dot{x} &= v_1, \\ \dot{y} &= v_2, \\ \dot{v}_1 &= u_1/m, \\ \dot{v}_2 &= u_2/m, \end{aligned} \tag{2.2}$$

or

$$\ddot{z} = u/m,$$

where m is the mass, $z=(x,y)^T \in R^2$ is the position, $u=(u_1,u_2)^T \in R^2$ is the input (force) and $(v_1,v_2)^T \in R^2$ is the velocity. The one above is also referred to as a *double integrator* in the plane, since the mapping from input to position involves integrating the input twice. Here it is also natural to bound the input force $\|u\| \le u_{\max}$.

Taking into account normalisation given in the new input $u' = u/m$ one can obtain $\ddot{z} = u'$ with an input bound $\|u'\| \le u_{\max}$.

This model was usually used for many types of robot, for example *omni-directional* robots, i.e. robots can freely and equally well move in all directions from the standing-still position. One of such real robot is the Nomadic XR4000 depicted in Figure 2.1.

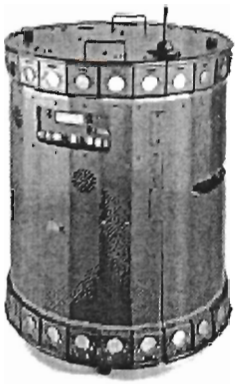


Figure 2.1: Nomadic XR4000 omni-directional mobile robot.

(Source: Global Dynamic Windows Approach Research
http://ai.stanford.edu/~oli/gdw.html, accessed May, 2006)

Models for the unicycle

The unicycle configuration is simple, therefore more common. It is the wheel geometry with two large independently actuated fixed wheels and one small free moving castor wheel to keep the balance as can be seen with Pioneer Robot in Figure 2.2 with its model schematically depicted in Figure 2.3.

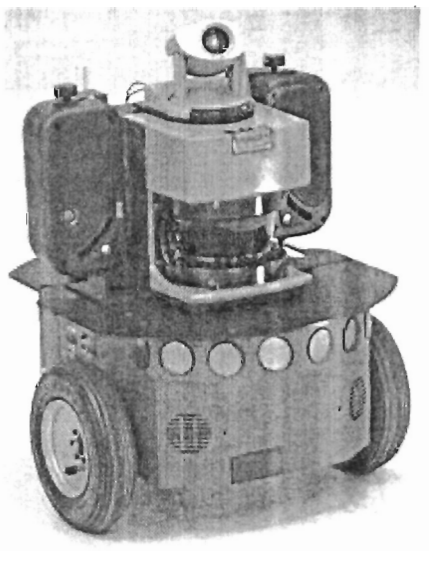


Figure 2.2: ActivMedia's Pioneer Robot P3-DX

(Source: ActiveMedia Robotics
http://www.activrobots.com/ROBOTS/p2dx.html, accessed May, 2006)

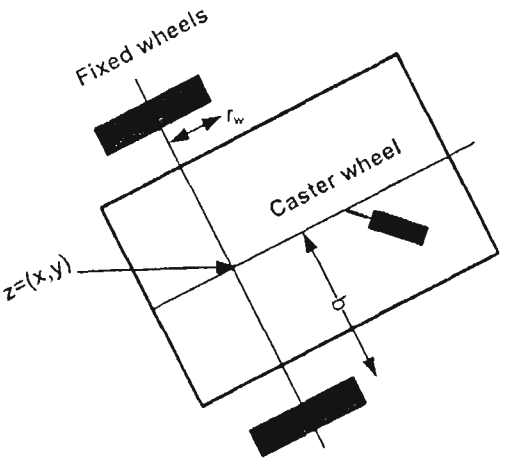


Figure 2.3: The unicycle model.

Unicycle kinematic model

The kinematic model for the unicycle is described as [Canudas de Wit *et al.*, 1996]

$$\begin{aligned} \dot{x} &= v\cos\theta, \\ \dot{y} &= v\sin\theta, \\ \dot{\theta} &= \omega, \end{aligned} \tag{2.3}$$

where $z=(x,y)^T$ is the centre point on the wheel axis, $\theta \in R$ is the orientation and input v,ω are the translational and angular velocities respectively.

This model describes many indoor robots as well as outdoor caterpillar-type and skid-to-turn vehicles, but for outdoor vehicles, the errors are bigger since the centre of rotation depends on the uncertain wheel-to-ground friction. It can also be used as a coarse model of an aeroplane if one adds bounds on angular velocity ω as well as a lower positive bound on translational velocity v. This renders a speed dependent minimal turning radius and is used to model unmanned aerial vehicles (UAVs) in [Beard *et al.*, 2002].

Throughout this thesis, this model is used with assumptions that robots satisfy pure-rolling and non-slipping conditions, which lead to non-holonomic velocity constraints:

non-slipping $$\dot{x}\sin\theta - \dot{y}\cos\theta = 0, \tag{2.4}$$

pure-rolling $$\dot{x}\cos\theta + \dot{y}\sin\theta = v\,. \tag{2.5}$$

This model is also called as *Knife edge model* since the behaviour of a mobile robot is similar to that of a thin knife. Some modifications of this model are *Roller unicycle model* and *Differential drive model.*

Roller unicycle model

This is a modification to the knife edge model that adds the rolling angle ϕ of the wheel as an additional degree of freedom. However, since the system can instantaneously possess only two degrees of freedom, there are two velocity level constraints, which can be represented as

$$\begin{aligned} &\dot{x}\sin\theta - \dot{y}\cos\theta = 0, \\ &\dot{x}\cos\theta + \dot{y}\sin\theta - r_w\dot{\phi} = 0, \end{aligned} \tag{2.6}$$

where r_w is the radius of the wheel. The first equation ensures that no sideways motion of the unicycle will exist while the second equation ensures that the unicycle will roll without slipping.

The model then can be described as

$$\begin{aligned} \dot{x} &= v\cos\theta, \\ \dot{y} &= v\sin\theta, \\ \dot{\theta} &= \omega, \\ \dot{\phi} &= v/r_w. \end{aligned} \tag{2-7}$$

Differential drive model

"Differential drive" refers the actuation of a mobile robot, which uses two independently driven wheels to manoeuvre in a plane like the movement of a conventional wheelchair. This model is an extension from the single knife edge model. The state variables are still $[x, y, \theta]$ but the inputs are the left wheel rotational velocity ω_1, and right wheel rotational velocity ω_2. The relationship between (ω_1, ω_2) and (v, ω) is

$$v = \frac{\omega_1 + \omega_2}{2} r_w,$$
$$\omega = \frac{\omega_2 - \omega_1}{2b} r_w, \tag{2.8}$$

where b is the half distance between the two wheels.

The model then can be described as

$$\dot{x} = \frac{1}{2} r_w \cos\theta(\omega_1 + \omega_2),$$
$$\dot{y} = \frac{1}{2} r_w \sin\theta(\omega_1 + \omega_2), \tag{2.9}$$
$$\dot{\theta} = \frac{1}{2b} r_w(\omega_2 - \omega_1).$$

Unicycle dynamic model

Adding two more states to satisfy Newton's law, the above kinematic model yields

$$\dot{x} = v\cos\theta,$$
$$\dot{y} = v\sin\theta,$$
$$\dot{\theta} = \omega, \tag{2.10}$$
$$\dot{v} = F/m,$$
$$\dot{\omega} = \tau/J,$$

where $z = (x, y)^T$ is the position, $\theta \in R$ is the orientation, v, ω is the translational and angular velocities, respectively, and the inputs are F/m and τ/J, in which F is the force, τ is the torque, m is the mass, and J is the moment of inertia of the robot.

Car-like vehicle model

A car-like mobile robot, for example Pioneer P3-AT in Figure 2.4, can be modelled schematically in Figure 2.5

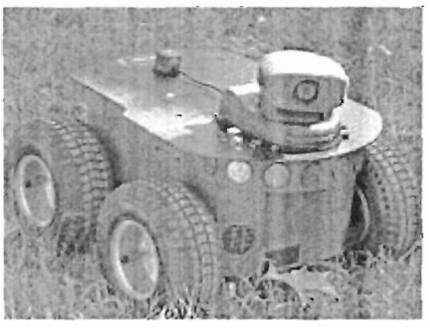


Figure 2.4: ActivMedia's Pioneer Robot P3-AT

(Source: ActiveMedia Robotics
http://www.activrobots.com/ROBOTS/p2at.html, accessed May, 2006)

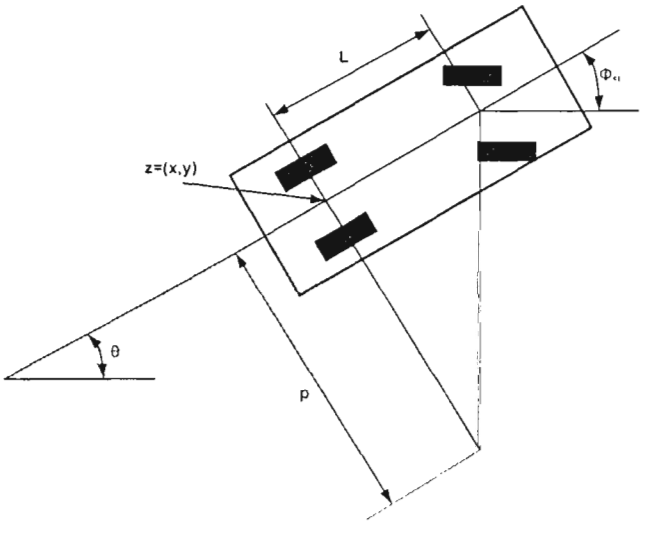


Figure 2.5: The car-type vehicle model

The back axis configuration resembles a unicycle and thus the $(x,y)^T$ equations are similar to (2.3). To understand the rotation control we assume the vehicle is moving under a constant steering angle ϕ_{st}. This makes the point $(x,y)^T$ trace out a circle with radius p, which in turn gives the angular velocity as $\dot{\theta} = \frac{v}{p} = \frac{v}{L}\frac{L}{p} = \frac{v}{L}\tan\phi_{st}$.

This gives the following equations:

$$\begin{aligned} \dot{x} &= v\cos\theta, \\ \dot{y} &= v\sin\theta, \\ \dot{\theta} &= \frac{v\tan\phi_{st}}{L}. \end{aligned} \tag{2.11}$$

There is a hardware bound on the steering angle, $\phi_{st} \in [-b_h, b_h]$. The appearance of v in all three equations excludes the possibility to turn on the spot as the unicycle. Newtonian dynamics can of course be incorporated in this model as well but it is rarely used in robotic research.

2.2 Control techniques for robotic formation

The robotic formation studies can be broadly classified into two groups [Erkin *et al.*, 2003]. The first group includes cases where the coordination is done by a centralised unit that supervises the whole group and commands the individual robots [Belta and Kumar, 2002; Egerstedt and Hu, 2001; Koo and Shahruz, 2001; Kowalczyk, 2002]. Assumptions about the existence of a central unit with a communication channel between it and individual robots make the centralised pattern formation strategies more costly, less robust to failures, and less scalable to the control of a large number of robots. Studies in the second group use distributed methods for achieving the coordination. This is the case when each control agent acts on the basic of local information and decisions [Carpin and Parker, 2002; Lawton *et al.*, 2003; Yamaguchi *et al.*, 2001].

While there are different approaches to the robotic formation control problem, the common theme remains the global coordination of multi agents to accomplish intelligently and/or autonomously some task objectives. Main design methods for distributed formation control under different scenarios can be grouped into four categories: behaviour-based and potential field, leader-follower, virtual structure, and other control strategies [Chen and Wang, 2005].

2.2.1 Behaviour-based and potential field approaches

Behaviour-based approach and potential field approach are usually combined in the application of formation control.

Suzuki and his colleagues developed a number of distributed formation algorithms [Sugihara and Suzuki, 1990; Sugihara and Suzuki, 1996; Suzuki and Yamashita, 1994].

In particular, they developed algorithms for multiple distributed mobile robots to form circles, simple polygons, and line segments; to uniformly distribute robots within a circle or a convex polygon; and to divide them into groups. Their research used models of idealised robots, which are represented by points, able to move in any direction, and equipped with range sensors that can determine the position of all other robots. The basic idea of their algorithms is to let each robot execute a simple algorithm iteratively and generate its new location adaptively at each iteration, based on a given goal and the positions of other robots.

In [Yun *et al.*, 1997] the behavioural approach is used to modify existing algorithms and propose some new algorithms for creating line and circle formations of a group of robots that are subject to physical constraints. The authors used Robot Simulator from Nomadic Technologies, Inc. Robots in Simulator realistically simulate the motion behaviour and sensor system of Nomas 200 mobile robots, which have a synchronous drive mechanism that enables them to translate, steer, and rotate (its turret) independently. The robot's sensor systems include tactile (bumper) sensors, infrared sensors, ultrasonic sensors, and laser sensors. Motion control and collision avoidance in the study are achieved by implementing a potential field algorithm. To each robot of concern, the presence of other robots generates a repulsive force that keeps them apart, and the goal position produces an attractive force. The proposed algorithms not only achieve the goal of forming a line or a circle; they also uniformly distribute robots on a line segment or a circle. Because the workspace is assumed to be obstacle-free, the shape of robots is circular, and the goal position changes as other robots move, the local minimum problem of the potential field method is rarely encountered in the simulation.

These ideas are extended in [Chen and Luh, 1994] to the problem of controlling a formation of mobile robots to transport objects. Once the robots have formed a desirable pattern in accordance with the object they are to transport, a control method is needed to coordinate the robots to maintain their geometric pattern while moving toward a given goal location, which is specified by some human operator. A "leader-following" coordination strategy was proposed. First, a suitable robot in the group is selected as the group leader. Once the leader is selected, the rest of the robots are considered as followers. During the course of transporting the object to its goal

location, the leader computes its next position at a given time interval, based on its relative position in the group and its knowledge about the goal location and orientation of the object. The leader broadcasts its anticipated position to the followers before it actually makes the move. Upon receiving the information regarding the leader's new position, the followers compute their corresponding new positions independently. They generate their motion plans by enforcing constraint satisfaction (object's orientation remains unchanged or changes as specified). The leader and followers proceed to their new locations after all the followers have completed their computations.

The behavioural approach is applied to the problem of maintaining a constellation of satellites in an equally distributed ring formation in earth orbit [McInnes, 1995]. Simple Lyapunov control functions are used to maintain distances and avoid collisions. The method uses information of the inter-satellite spacing to generate low-thrust radial and transverse control accelerations. Using the concept of potential functions, the uniform ring is seen as a minimum energy configuration of the system. The control accelerations ensure that the potential function of the entire system monotonically decreases so that this minimum energy configuration is achieved from any initial configuration. The application of the behavioural approach to aircraft flying in formation is described in [Anderson and Robbins, 1998], where the control strategies are derived to mimic the instinctive behaviour of birds and fish. The aircraft in the formation are allowed to manoeuvre individually. However, the manoeuvre chosen by each aeroplane stems from basic flocking instincts such as collision avoidance, obstacle avoidance, and formation-keeping.

In [Balch and Arkin, 1998] each robot has basic motor schemas. Each schema generates a vector representing the desired behaviour response to sensory input. Possible motor schemas include collision avoidance, obstacle avoidance, goal seeking, and formation keeping. The control action of each robot is a vector weighted average of the control for each motor schema behaviour. Unit-centre tracking, leader tracking and nearest neighbour tracking controls are also studied. Since competing behaviours are averaged, occasionally strange and unpredicted behaviours may occur.

In [Cao *et al.*, 2003] five primitive behaviours (move-to-goal, keep-formation, avoid-obstacle, avoid-robot, and random) with a motion prediction model of moving obstacles

are adopted. Parabola prediction model is used to predict the positions of moving obstacles and its parameters are estimated by the recurrence least square algorithm with restricted scale in case of datum saturation. The authors also designed a series of generation functions to generate control parameters for behaviours' combination. Relying on that, the adaptability of robots' motion to the environment is improved. In another work, Cao *et al.* (2002) used Genetic Algorithm, which is widely known as global optimization technique, to decide the control weights and choose the appropriate behaviour for formation maintenance and obstacle avoidance.

In [Monteiro and Bicho, 2002], the behaviour-based formation control is modelled into a non-linear dynamic system for trajectory generation and obstacle avoidance for a team of three autonomous robots. In this approach, the level of modelling is at the level of behaviours. A "dynamics" of behaviour is defined over a state space of behavioural variables. The environment is also modelled in these terms by representing task constraints as attractors (i.e. asymptotically stable states), or repellers (i.e. unstable states) of behavioural dynamics. For each robot, attractors and repellers are combined into a vector field that governs the behaviour. Simulation results show that the generated trajectories are smooth. Flexibility is achieved in that as the sensed world changes, the systems may change their planning solutions continuously but also discontinuously (tunning the triangle formation versus split to avoid obstacles).

Fredslund and Mataric (2002) used local information to establish and maintain formations of a group of mobile robots. Each robot in the group has a unique ID and a designated friend robot, which it can see through a "friend sensor". For that purpose, each robot has a coloured cylinder on its structure so that other robots can recognise it. In addition, a laser range finder is used to infer distance between robots. There is a minimal communication between robots: heartbeat signals (i.e. robot broadcast their IDs), swerve signals (changing direction to avoid obstacle), and formation messages. Each robot can learn the number of robots in formation and the type of formation using broadcast messages. In addition, for each formation, each robot has a locally calculated angle, which determines the angle it should keep between its front direction and the direction of its friend. This study accomplishes the task of forming and maintaining formations using only local information and minimal communication. The possible

formations however are limited to chain-shaped ones and cannot make a backward curve. Switching formation is possible, but may require intermediate repositioning depending on the formation to switch to. Also, if the robots in the group are initially in a random position, the time required to initialise the formation shape may not to be optimal.

Dudenhoeffer and Jones (2000) described a flexible architecture for modelling thousands of autonomous agents simultaneously. Using this simulation tool, the problem of hazardous material detection by thousands of micro-robots scattered around a region is tackled. The agents' behaviour is based on a subsumption architecture in which individual behaviours are prioritized with respect to all others. The primary behaviour explored in this work is a group formation behaviour based on social potential fields [Reif and Wang, 1999]. Robots are desired to stay at specific distances from others to obtain optimum coverage of the area. They are also required to wander in this formation to search other parts. In this research, the social potential field method is extended and evaluated in the presence of agent failure and imperfect sensory input. The method uses only local information and is scalable to very large groups of robots.

In [Ge *et al.*, 2004], the desired formation pattern and trajectory for the robot group are presented by artificial potential tranches. First, formations are defined using the concept of queues instead of nodes. This supports a large variety of different formations and allows for the automatic scaling of formations according to changes in the overall size of the robot team. A decentralised redistribution algorithm is used, which enables the robots to redistribute themselves dynamically and efficiently amongst queues. Each robot uses information gathered from the broadcast channel and the robot's local sensors. Artificial Potential Trenches, each associate with a queue, are then used. Each robot will be attracted and move along the bottom of the "valley" created by the potential field; and robots automatically distributed themselves along the trench. This method addresses the issues of scalability, flexibility and stability, which are required for the formation control of a large team of networked homogeneous robots with explicit communication capabilities. In addition, the obstacle avoidance behaviour considers the relative position and velocity, thus taking into account the presence of

moving obstacles. Local minima problems due to deep crevices of obstacles and dead end of narrow paths, are solved by the application of the instant goal method.

In [Yamaguchi and Arai, 1994] a distributed and autonomous control method called Linear Autonomous System (LAS) is proposed for generating a shape of a group that consists of multiple mobile robots. In this method, a decision of shape of the group is attributed to designing a potential field that is spread on a space of relative distances between mobile robots. With the potential filed, controllability of group shape can be defined. Because each robot has parameters for designing the spread potential field, a shape of the group can be adapted to various situations by means of changing the parameters with respect to state of environment, for example, the position of a wall, a velocity of an obstacle and so on.

Balch and Hybinette (2000a, 2000b) proposed a potential approach for robot formation that is inspired from the way molecules form crystals. In this study, each robot has several local attachment sites that other robots may be attracted to. This concept is similar to molecular covalent bonding. Possible attachment site geometries include shapes resembling 'X','|', or '+', where the robot is the centre of the shape and the attachment sites are the ends of the line segments. Various robot formation shapes result from usage of different attachment site geometries just as different crystal shapes emerge from various covalent bond geometries. When a team of robots moving in a formation, they avoid an obstacle by splitting around the obstacle and rejoining after passing it. This approach is scalable to large robot teams since global communication is not used and that local sensing is sufficient to generate effective formation behaviours in large robot teams.

The analysis of the above research papers shows the advantages and disadvantages of behaviour-based and potential field approaches. Their advantage rests in that it is naturally intuitive to derive control strategies when agents have multiple competing objectives. In addition, there exists an explicit feedback to the formation since each agent reacts according to the position of its neighbours. Another advantage is that the behavioural approach is naturally implemented in a decentralised manner. The primary disadvantage is that group behaviours cannot be explicitly defined, rather a group behaviour is said to "emerge". Another weakness is that behavioural approaches are

difficult to analyse mathematically and characteristics of formation (like stability) cannot generally be guaranteed.

2.2.2 Leader-follower approaches

In leader-follower approaches, one or more agents are designated as leaders, with the rest of the control agents designated as followers. The basic idea is that the followers track the position and orientation of the leaders with some prescribed (possibly time-varying) offsets.

One of the first studies on leader-following strategies is the work of Wang (1991) which discusses formation control laws for mobile robots. In this study, each robot is represented by a particle with a spherical effective spatial domain and a specified cone of visibility. The global motion of each robot in the world space is described by the equations of motion of the robot's centre of mass. First, methods for formation generation are discussed. Then, simple navigation strategies (nearest-neighbour tracking, multi-neighbour tracking, inertially referenced movements, and mixed nearest-neighbour tracking and inertially referenced movements) for robots moving in formation are derived. These strategies have features similar to those used by human in steering land vehicles and aircrafts in formations. A sufficient condition is obtained for the stability of the formation pattern for a fleet of robots, each equipped with a particular type of navigation strategy. Simulation results show that the strategies based on nearest-neighbour tracking are effective when inter-robot communication and complete visibility of the neighbouring robot are maintained at all times. However, collision-avoidance strategies have not been incorporated with the strategies for moving in formations derived in the research.

The application of these ideas to spacecraft formations is described in [Wang and Hadaegh, 1996], where explicit control laws for formation keeping and relative attitude alignment based on nearest neighbour tracking are derived. Each microspacecraft is modelled by a rigid body with fixed centre of mass. Several leader-following techniques are discussed including leader tracking, nearest neighbour tracking, barycenter tracking, and other tree topologies. The necessary data, which must be communicated between microspacecraft to achieve the effective control, are also

examined. The paper concludes with a discussion of the implementation of the derived control laws, and the integration of the microspacecraft formation coordination and control system with a proposed inter-spacecraft communication/computing network.

In [Wang *et al.*, 1999], these ideas are extended to account for actuator saturation and are applied to the problem of controlling the formation to execute a continuous rotational slew. Using a particle model for spacecraft formation dynamics and a rigid-body model for spacecraft attitude dynamics, control laws are derived for rotating the entire formation about a given axis and synchronising individual spacecraft rotation with formation rotation in the absence of a gravitational field and disturbances. A simplified control law suitable for implementation is also obtained. This shows that under mild conditions the formation alignment error decays to zero exponentially with time. In [Hadaegh *et al.*, 1998], adaptive control laws are added to the control derived in [Wang and Hadaegh, 1996] in order to reject common space disturbances.

Spacecraft control using the leader-following concept is reported in [De Queiroz *et al.*, 2000] for keeping satellite formation in earth orbit. Idealised scenario, where the spacecraft actuators are capable of providing continuos-time control efforts, as opposed to being of pulse-type, is considered. Specifically, the full nonlinear dynamics describing the relative positioning of multiple spacecraft formation flying is used to develop a Lyapunov-based, nonlinear, adaptive control law that guarantees global asymptotic convergence of the position tracking error in the presence of unknown, constant, or slow-varying spacecraft masses, disturbance forces, and gravity forces. In the case when the parameters are exactly known, the proposed control strategy yields global exponential convergence of the tracking errors.

In [Sugar and Kumar, 1998], the leader-follower strategy is used to control a group of mobile robot to cooperatively move a box. First, the experimental prototype of a mobile manipulator that consists of off-the-self mobile platforms equipped with a novel, three degree-of-freedom parallel manipulator is described. Then in this proposed architecture for coordinated control of multiple mobile platforms, a lead robot plans, based on available sensory information, and follows a suitable trajectory. The other robots follow a desired formation with respect to the leader while maintaining a stable

grasp. The wireless network required for communication and rapid prototyping of real-time control code is also presented.

Feedback linearization techniques are used in [Desai, 2001; Desai, 2002; Desai *et al.*, 1998; Desai *et al.*, 2001] to derive tracking control laws for non-holonomic robots in a formation that is described as a directed graph. The authors presented the terms $l-\psi$ and $l-l$ control to reflect whether the control laws are based on tracking the position and orientation of the robot relative to a leader, or the positions relative to two leaders, respectively. The shape of the formation is changed as graph structures are changed.

Three controllers, namely *separation-bearing*, *separation-separation*, and *separation distance-to-obstacle* , in which first two are adopted from [Desai *et al.*, 1998], are used in [Fierro *et al.*, 2001; Fierro *et al.*, 2002] to allow robots to control their positions and orientations with respect to neighbouring robots or obstacles in the environment. The authors also outlined a coordination protocol for automatically switching between control laws to maintain a specific formation. Two simple trajectory generators, which are derived from potential field theory, are proposed. The first allows each robot to plan its reference trajectory based on the information available to it. The second scheme requires sharing of information and enables a rigid group formation. The ideas are extended in [Das *et al.*, 2002] with estimators that abstract the sensory information at different levels, enabling both decentralised and centralised cooperative controls. The approach is vision-based with each robot identified by a colour using omnidirectional cameras. Because the formation, its leader and the allowed switch between formations must be predetermined, the approach cannot be optimal depending if the environments are known or not, or if it is possible or not to initialise the positions of the robots in a good configuration for the desired formation.

In [Takahashi *et al.*, 2004] the concept of performance index that shows mobile robot ability is presented. Specifically, maximum acceleration and maximum velocity of a robot are defined by maximum admissible rotation and maximum continuous torque of a motor. The performance index is quantified from arrival time on the destination using the parameters. Based on that, new $l-\phi$, $l-l$ controllers are suggested along with a compliance controller using a virtual repulsion to avoid robots colliding with each other.

A real-time system of multiple mobile robots with RT-Messenger is used as a platform for simulations and experiments. RT-Messenger allows robots to transmit information regarding their positions to each other in real time.

Yamaguchi and Burdick (1998) introduced a smooth time-varying feedback control law to form group formations for multiple Hilare-type mobile robots. To control the formation, each robot has its own coordinate system and it controls its relative positions to its neighbouring robots. Particularly, it has a vector called "a formation vector", and the formation is controllable by the vectors. Its asymptotic stability is guaranteed in mathematical framework, averaging theory.

A leader–follower approach applied in intelligent highways is addressed in [Sheikholeslam and Desoer, 1992]. The overall system consists of N vehicles (the platoon) where each vehicle is driven by the same input u and the state of the kth vehicle affects the dynamics of the $(k+1)$th vehicle. Furthermore, the dynamics of each vehicle is affected by its (local) state–feedback controller. This work proves that under some general qualitative conditions on the dynamics of vehicle models, decentralised controllers can achieve the design goals of the platoon concept: multiple vehicles travelling down the highway at high speed and maintaining tight formations.

The investigation on the stability properties of mobile agent formations which are based on leader following is presented in [Tanner *et al.*, 2004]. The authors derived nonlinear gain estimates that capture how leader behaviour affects the interconnection errors observed in the formation. Leader-to-formation stability (LFS) can be used to gain quantify error amplification, relate interconnection topology to stability and performance, and offer safety bounds for different formation topologies. The notion is based on input-to-state stability and its invariance properties under cascading. The intuitive fact that performance deteriorates as the graph that represents the formation interconnections increases in diameter, can now be formally justified.

The strength of leader-follower approaches is that group behaviour is directed by specifying the behaviour of the leaders. The weakness, however, is that there is no explicit feedback to the formation. In some cases, for example, the leader may be moving too fast for the follower to track. Another weakness is that the leader is a single

point of failure for the formation. The formation convergence depends on each follower knowing the position, velocity and sometimes even acceleration of the leader. If this information is not available, then no convergence can be guaranteed. This implies that a large communication burden is usually necessary for leader-follower approaches. In addition, due to communication latency and communication delays each robot will have to figure out what the leader is doing with old information that it receives only every couple of seconds.

2.2.3 Virtual structure approaches

The concept of virtual structure was firstly introduced in [Lewis and Tan, 1997; Tan and Lewis, 1996]. The entire formation is treated globally as a single structure or so-called virtual structure. If the desired dynamics of the virtual structure can be translated into the desired motion of each robot then one can design local controllers to achieve global performance. This approach is illustrated in Figure 2.6 with four stpes. In step 1 of the figure, the robots are situated in a triangular virtual structure. A virtual force field moves the structure (step 2), and then the robots compute trajactories and reposition themselves with respect to the structure (step 3 and 4). The flow of control is actually bidirectional: movement of the virtual structure causes the robots to reposition themselves; movement of the robots can cause the virtual structure to reposition.

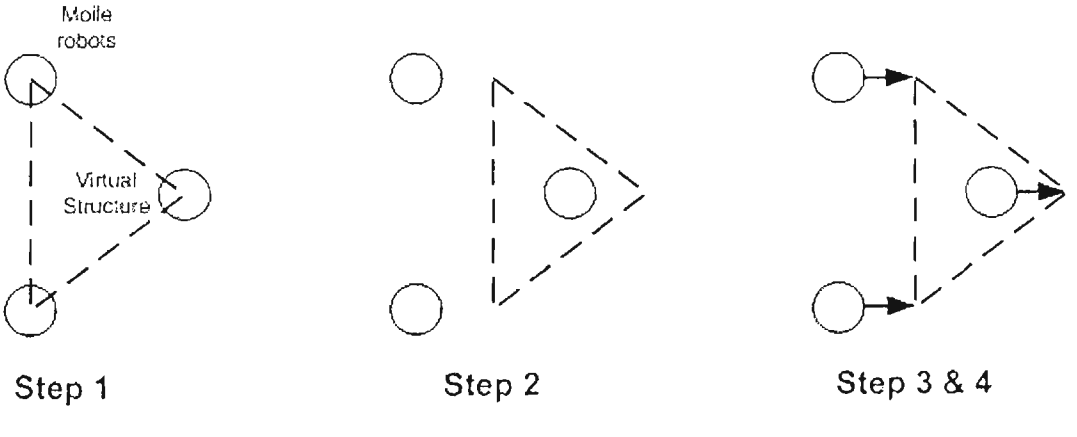


Figure 2.6: Steps in the virtual structure control algorithm

The application of virtual structure techniques to formations of spacecraft in free space is described in [Beard, 1998; Beard and Hadaegh, 1998]. The basic architecture used in their research approaches is that of a constellation template, i.e. a virtual rigid body,

with an inertial position and orientation that specifies the desired position and attitude of each spacecraft within the constellation. Some algorithms are derived for four basic constellation manoeuvres: reorientation, rotation, expansion/contraction and initialisation. A constellation reorientation manoeuvre requires the entire constellation to change orientation while the individual spacecraft in the constellation maintain their relative positions and orientations with respect to each other. A constellation rotation manoeuvre is similar to a reorientation manoeuvre, except the constellation is rotated at a constant rate about an axis in the constellation. A constellation expansion or contraction manoeuvre requires the volume of the convex hull of the spacecraft to uniformly increase over some time interval. Constellation initialisation is requiired after lauch of the spacecraft. The difficulty with initialisation is that the spacecraft may have a limited inter-spacecraft sensing and communication ability before they form into the desired constellation.

The virtual approach can be implemented as a leader-follower control. We can treat the N robots in the formation as followers and then designate one "virtual" robot as a leader. If one chooses the dynamics of the virtual robot to evolve as a second order system, the way that formation envolves can be chosen over time by appropriate picking the system parameters of the virtual leader.

The coordination architecture that combines a virtual structure method with leader-follower method and behaviour-based approach to formation control of multiple spacecraft interferometer in deep space is presented in [Beard *et al.*, 2001]. In this study, to achieve global coordination, knowledge of the virtual structure states is shared between each agent through dynamic coordination variables. Note that these variables are similar to the action reference notion introduced in [Kang *et al.*, 2000] or the platoon-level functions given in [Stilwell, 2002]. This reserach demostrates the application of the proposed architecture to the problem of synthesizing a deep-space, free-flying, multiple sapcecraft interferometer. A constellation of three spacecraft will firts be initialised into a formation. The formation will then be retargeted to point at a star. Next, the formation will be controlled to cover several U-V interferometer points. A high-precision station keeping manoeuvre is then perfomed at each U-V point.

A similar idea is applied in [Ren and Beard, 2004], where, decentralised formation control strategies are introduced, which are appropriate when a large number of spacecraft are involved and/or stringent inter-spacecraft communication limitations are exerted. Each spacecraft in the formation instantiates a local copy of the formation control, i.e. the coordination vector, and the local instantiation of the coordination vector in each spacecraft, which represents the states of the virtual structure, is synchronized by infrequent communication with its neighbours following a bidirectional ring topology.

The strength of the virtual is that it is fairly easy to prescribe a coordinated behaviour for the group. Futhermore, feedback to the virtual structure is naturally defined. The disadvantage is that requiring the formation to act as a virtual structure limits the class of potential applications of this approach. Another weakness is that the virtual structure approach is naturally implemented in a centralised manner. This increases the communication burden to a higher order beyond that of the leader-follower approach.

2.2.4 Other control strategies

Generalised coordinates

In [Spry and Hedrick, 2004], a control methodology based on generalized coordinates is presented. These coordinates characterise the vehicle's location (L), orientation (O) and its shape (S) with respect to a formation reference point in the formation. The location of the formation is defined as the location of a formation reference point (RFP). The orientation of the formation is defined as the orientation of a formation reference frame (FRP). The shape of the formation is defined as its configuration relative to the FRP. The trajectories of the formation group can be specified in terms of L, O and S coordinates. Both force-based and velocity-based controls then are developed for asymptotic tracking of trajectories while maintaining a desired formation geometry. A similar idea is presented in [Yamakita and Saito, 2004], where a control method for formation control of child robots in a multiple mobile robot system called Super Mechano Colony (consists of a single mother ship and multiple mobile robots that have a parent-children relationship) is proposed using multiple coordinate systems, i.e. physical coordinate system and shape coordinate system. Decentralised control in a

physical coordinate system and centralised control in a shape coordinate system are introduced and the combined control is proposed. The properties of the combined control method are shown by numerical simulations and experiments.

In [Zhang *et al.*, 2003] formations that contain a small number of robots are modelled as controlled Lagrangian systems on Jacobi shape space. This allows a block-structured control of position, orientation and shape of the formation. Feedback control laws are derived using control Lyapunov functions. The controlled dynamics converges to the invariant set where desired shape is achieved. Controllers are implemented in a layered fashion via the extended motion description language (MDLe) system. Group MDLe plans are constructed to allow structured controller design for formations.

Nonlinear servomechanism

Gazi (2003) showed that under some assumptions on the trajectories to be tracked by the formation and the local formation dynamics, the problem of agents moving in a formation can be approached in the framework of nonlinear output regulation (servomechanism). The author considers a generic model for an agent with general nonlinear dynamics. The agents are required to follow a virtual leader, determined by a set of tracking constraints, and to keep a required formation, determined by a set of formation constraints. The limitation of the framework is that the dynamics of the virtual leader are required to be generated by a neutrally stable autonomous system. Nevertheless, the class of reference trajectories that could be tracked is still large enough and with practical interest (such as the trajectories of clusters of orbiting satellites).

Genetic Algorithm and Reinforcement Learning

A switching algorithm between GA (Genetic Algorithm) based formation control and RL (Reinforcement Learning) based obstacle avoidance is proposed for re-establishing the formation of a multi-robot system in an environment with obstacles [Kobayashi *et al.*, 2003]. Each robot acquires its control rule for establishing the formation according to neighbouring robots by GA and the action policies for avoiding obstacles by RL. Then, each robot can switch between the formation control and obstacle avoidance

according to own sensing information. The proposed method is evaluated in simulations.

Model Predictive Control

In [Yan and Bitmead, 2003] Model Predictive Control (MPC) is used as a localised control law to connect the overall formation control performance and the inter-vehicle communication quality. The inaccurate inter-vehicle communication is modelled as white noise. The working problem is a simple 1-D vehicle formation with noisy channels. The measure of information quality is the covariance of the state estimates. The interaction of the information quality in terms of estimate covariances with the formation performance is investigated under this framework. The modified version of MPC controller can incorporate the information quality into the control law and keep the same framework while investigating different information structures with different quality.

Geometric and dynamic tasks

In [Skjetne *et al.*, 2002], for the control of a group marine-craft, the formation maintenance and the trajectory tracking problem are decomposed into a geometric task and a dynamic task. The geometric task ensures that the individual ship converges to the position in the formation while the dynamic task will make sure that the ships travel along the trajectory with the desired speed. The formation control problem is solved by introducing a formation reference point (FRP) and designating each vessel a relative position to that point. Using the manoeuvring design, the geometric and the dynamic tasks ensure that the FRP converges to and follows a desired path $\xi(\theta)$ with a specified speed, where θ is the path parametrization variable. The desired motion of the FRP is based on the states of all the vessels in the formation. The drawback here is the centralised update law for $\dot{\theta}$ which needs full state information from all vessels in the formation. Hence, for r vessels, each with n states, the number of communicated signals is $rn+1$. In order to reduce the signal flow, the formation control problem was solved as a decentralised scheme in [Skjetne *et al.*, 2003] by solving an individual manoeuvring problem for each vessel with an individual path variable θ. Then, by

synchronizing all the θ_i, $i = 1,2,...$ the formation, represented by the FRP, will move along the path ξ with the desired speed. Similar idea can be found in [Lee and Li, 2003] where a decomposition of the dynamics of multiple spacecraft includes two parts: an average system which represents the overall motion of the group and a shape system which governs the group formation structure. In addition, the decomposition has the property that the sum of the energies of the average and the shape system equals the energy of the original system. Thus, by designing suitable control laws for the average and the shape system respectively, desired group manoeuvre and internal formation are achieved.

Vision-based Follow-the-Leader

In [Cowan *et al.*, 2003], the leader-follower approach is used for formation while the navigation function is used for collision avoidance for the formation. The authors derived the equations of motion of the leader in the image plane of the follower and proposed two control schemes for the follower, which are either, string, leader-to-formation, or locally stable depending on the sensing capabilities of the followers. The first one is based on feedback linearization. The second one assumes a kinematic model for the evolution of the leader velocities and combines a Luenberger observer with a linear control law.

Fuzzy logic and neural network

Hong *et al.* (2001) proposed an architecture of fuzzy system for each robot speed control and fuzzy-neuro system for obstacle avoidance in a formation. The controller adopts a simple reactive navigation strategy by combining repulsion from obstacles with attraction to a goal. Robots can maintain formation with respect to unit-centre. Simulation results show that the proposed strategy is effective for multi-robot to avoid obstacles while maintaining a formation.

Visual servoing

In [Cowan *et al.*, 2003; Tan and Lewis, 1996; Vidal *et al.*, 2003], the visual servoing problem is considered for the application of formation control. Tan and Lewis (1996)

used a vision-based tracking system (called V-GPS: Vision-based Global Positioning System) to monitoring the position of each robot. Vidal *et al.* (2003) considered a formation control scenario in which motion segmentation techniques enable each follower to estimate the image-plane position and velocity of the other robots in the formation, subsequently used for omnidirectional image-based visual servoing. Cowan *et al.* (2003) considered non-holonomic robots equipped with central panoramic cameras, assume a desired formation in the image plane and use omnidirectional visual servoing for tracking. Moreover, flocking of mobile agents is investigated in [Tanner *et al.*, 2003a; Tanner *et al.*, 2003b], which consider the stability properties of a system of multiple mobile agents with double integrator dynamics. In one paper, the topology of the control interconnections between the agents in the group is fixed and time invariant. Each agent regulates its position and orientation based on a fixed set of "neighbour". The other paper considers the topology that varies with time, i.e. the set of neighbours may change in time, depending on the relative distances between the agent and its flockmates. Flocking can be established when each agent reacts only to flockmates within a limited neighbourhood around itself.

The advantage of these techniques in formation control is that they are usually mathematically analysed and formation stability can be guaranteed in the most of research. Techniques in artificial intelligence promise applications in robotic formation, which is inspired from natural world. However, research in this category usually validates its techniques by computer simulations and concerns only some of issues of formation control problem under some assumptions. Common models of mobile robots are rarely considered to derive control laws for real robots.

2.3 Conclusion

This chapter has reviewed some most common mathematical models of mobile robots used in robotic research. Different models of ground-based mobile robots are used in robotic research depending on real platforms for practice. Control techniques for robotic formation are also surveyed, highlighting their advantages and disadvantages.

The combination of these techniques may be required for practical scenarios of robotic formation.

In the literature, although there are many different methods proposed for formation control, the robotic formation problem, especially the initialization issue has not been adequately addressed. It is one of the main objectives of this thesis to contribute to this research topic.

Chapter 3

Virtual head robot tracking control

3.1 Introduction

Wheeled mobile robots (WMRs) have been an active area of research and development over several decades. Based on the wide range of applications of them in large (potentially unstructured and hazardous) domains, it is clear that WMR research is multidisciplinary by nature. For control of a WMR, there are two main problems: regulation problem that considers how to force the actual position and orientation of a WMR to a constant reference position and orientation, and tracking problem, whose objective is to force the actual position and orientation of WMR to track a time-varying reference trajectory. Several controllers have been proposed for the regulation and reference robot tracking problem, especially Dixon *et al.* (2001) proposed a new class of unified differentiable, time-varying kinematic controllers to address both these problems simultaneously. These existing controllers consider the control problem of a single mobile robot.

In the scenario of multiple mobile robot coordination, as reviewed in Chapter 2, approaches to formation control of a group of agents can be fundamentally categorized into four board groups: behaviour-based and potential field, leader-follower, virtual structure, and other control strategies. There are two main problems in leader-follower approaches: leader tracking and collision avoidance. For tracking, Desai *et al.* (1998)

proposed two types of feedback controllers for maintaining formations of multiple mobile robots: *l-ψ* controller and *l-l* controller. Another approach used three types of controllers (*basic leader-following*, *leader-obstacle*, and *three-robot shape*, which are also called *separation-bearing*, *separation distance-to-obstacle*, and *separation-separation)* to maintain a formation under appropriate assumptions on the motion of the lead robot [Das *et al.*, 2002]

In the above approaches, a simple leader-follower configuration is obtained by using a basic *l-ψ* or *separation-bearing* controller. *Virtual Robot Tracking* (VRT) control proposed in [Jongusuk and Mita, 2001] could be also used for that purpose with the consideration of inter-robot collision avoidance. However this approach has limitations in the establishment of a line configuration and the possibility of collisions between robots in some cases. The *Virtual Head Robot Tracking* (VHRT) control will be proposed in this chapter to overcome these circumstances.

Section 3.2 presents models and problem formulation. An overview of $l-\psi$, *separation-bearing*, and *virtual robot tracking* control is given in Section 3.3. To overcome the limitations of *virtual robot tracking* control, a new VHRT control is proposed. Details of the control design are presented in Section 3.4. Section 3.5 presents some simulation results to validate the proposed approach. A conclusion is given in Section 3.6.

3.2 Model and problem formulation

3.2.1 A nonholonomic mobile robot model

A three-wheel mobile robot can be described by a common kinematic model as presented in Chapter 2:

$$\begin{aligned} \dot{x} &= v\cos\theta, \\ \dot{y} &= v\sin\theta, \\ \dot{\theta} &= \omega, \end{aligned} \tag{3.1}$$

where $z=(x,y)^T$ is the centre point on the wheel axis, $\theta \in R$ is the orientation and input v,ω is the translational and angular velocities respectively. The nonholonomic velocity constraints require that robots must satisfy strictly pure rolling and non-slipping conditions.

3.2.2 Collision detection model

A simple technique used to detect the collision between any two robots can be applied by modelling each robot to be covered with a circle centering at its control point on the wheel axis. The radius r_{safe} of that circle can be determined by taking the real robot's dimensions into account with an additional distance as a safety margin. The distance ρ between two robots can be denoted as the distance between their control points (see Figure 3.1).

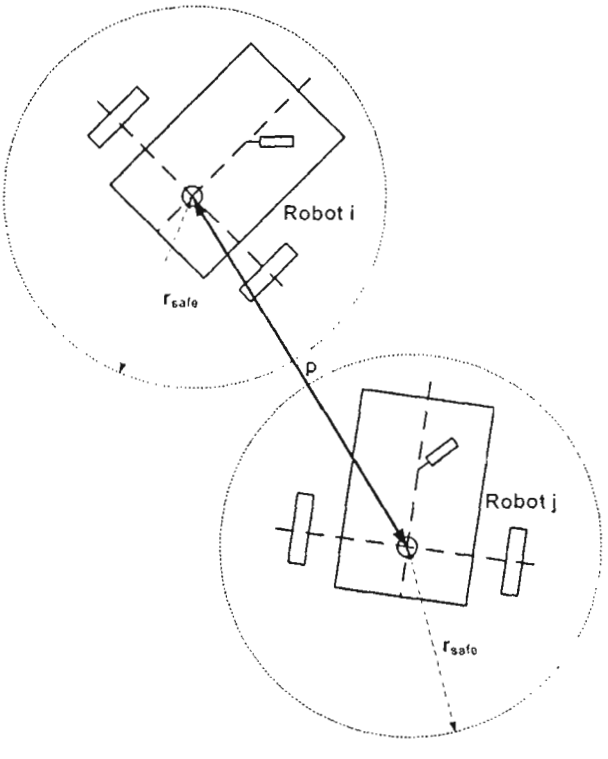


Figure 3.1: Collision detection

Consider two mobile robots, which are indexed by i and j respectively. A measure describing the possibility of collision between them can be written as

$$f_{ij} = \rho - 2r_{safe}. \tag{3.2}$$

The value of f_{ij} represents the possibility of collision between two robots:

$$\begin{aligned} f_{ij} > 0 \quad &\rightarrow safe, \\ f_{ij} \leq 0 \quad &\rightarrow unsafe. \end{aligned} \tag{3.3}$$

3.2.3 Assumptions

Let us introduce some assumptions:

- **Assumption 1**: Two robots (leader i and follower j) are of the same model as described in (3.1) and strictly satisfy pure rolling and non-slipping conditions. The possibility of collision between them can be assessed using function f_{ij}.

- **Assumption 2:** Follower j can get any necessary information about its position and orientation and information of leader i in a global coordinates from its communication channel.

- **Assumption 3:** The leader follows a smooth trajectory and the workspace is flat and obstacle-free.

3.2.4 Problem statement

The objective here is to design a controller for follower j to achieve:

1. any desired configuration between leader i and follower j, and

2. no collision between two mobile robots.

3.3 l-ψ control, separation-bearing control, and virtual robot tracking control

This section describes and analyses some existing control laws for robot tracking to form a desired leader-follower configuration of two mobile robots.

3.3.1 *l*-ψ control

This controller is originally proposed by Desai *et al.* (1998) to maintain a desired length l_{ij}^d and a desired relative angle ψ_{ij}^d between two robots.

System model

Figure 3.2 shows a system of two nonholonomic mobile robots separated by a distance of l_{ij} between the centre of robot i and the front castor of the robot j. In this system, each robot has two actuated degrees-of-freedom and two robots are not physically coupled in any way. The distance between the castor and the centre of axis of the wheels of each robot is denoted by D. Let us denote the coordinates of the centres of axis of the wheels of robot i and robot j are (x_i, y_i) and (x_j, y_j) respectively. The orientations of robot i and robot j are θ_i and θ_j, respectively.

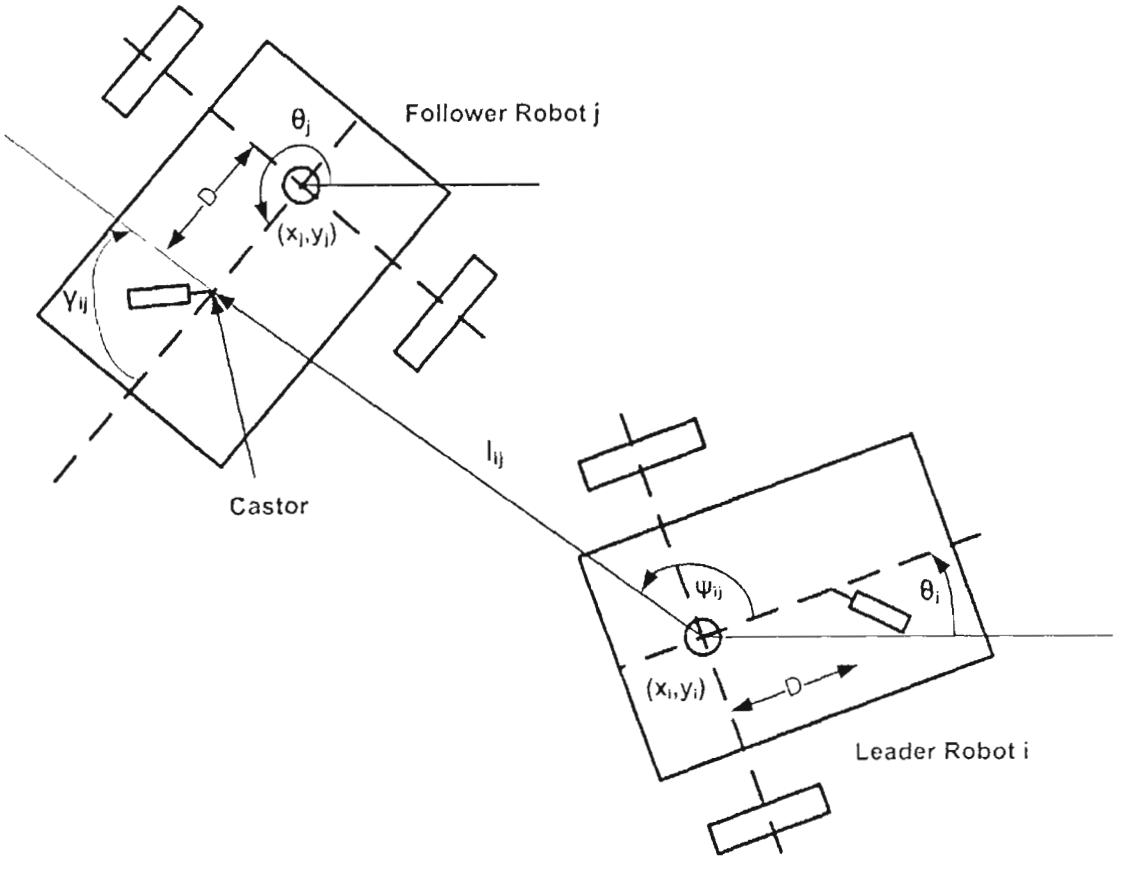


Figure 3.2: Notation for *l*-ψ control

The coordinates of the castor of robot j then can be written as

$$\begin{aligned} x_{cj} &= x_j + D\cos\theta_j, \\ y_{cj} &= y_j + D\sin\theta_j. \end{aligned} \tag{3.4}$$

Position errors between the castor of robot j and the centre of robot i are

$$\begin{aligned} \Delta x_{ij} &= x_{cj} - x_i = x_j + D\cos\theta_j - x_i, \\ \Delta y_{ij} &= y_{cj} - y_i = y_j + D\sin\theta_j - y_i. \end{aligned} \tag{3.5}$$

It is clear from the notation in Figure 3.2 that

$$\begin{aligned} \Delta x_{ij} &= l_{ij}.\cos(\psi_{ij} + \theta_i), \\ \Delta y_{ij} &= l_{ij}.\sin(\psi_{ij} + \theta_i), \end{aligned} \tag{3.6}$$

and

$$l_{ij}^2 = (\Delta x_{ij})^2 + (\Delta y_{ij})^2. \tag{3.7}$$

Differentiating equation (3.7) with respect to time, one can obtain

$$l_{ij}.\dot{l}_{ij} = \Delta x_{ij}.\dot{\Delta} x_{ij} + \Delta y_{ij}.\dot{\Delta} y_{ij}. \tag{3.8}$$

The derivatives of position errors $\dot{\Delta} x_{ij}$ and $\dot{\Delta} y_{ij}$ can be obtained from (3.5) and model (3.1):

$$\begin{aligned} \dot{\Delta} x_{ij} &= v_j\cos\theta_j - D\omega_j\sin\theta_j - v_i\cos\theta_i, \\ \dot{\Delta} y_{ij} &= v_j\sin\theta_j + D\omega_j\cos\theta_j - v_i\sin\theta_i, \end{aligned} \tag{3.9}$$

where (v_i, ω_i) and (v_j, ω_j) are the linear and angular velocities at the centres of axle of robot i and robot j, respectively.

By substituting $\Delta x_{ij}, \Delta y_{ij}$ from (3.6), $\dot{\Delta} x_{ij}, \dot{\Delta} y_{ij}$ from (3.9) into (3.8) and taking some simple manipulations, one can obtain the kinematic equation of the distance between two robots as follows.

$$\dot{l}_{ij} = v_j\cos\gamma_{ij} - v_i\cos\psi_{ij} + D\omega_j\sin\gamma_{ij}, \tag{3.10}$$

where $\gamma_{ij} = \theta_i + \psi_{ij} - \theta_j$.

Similarly, from (3.6), one has

$$\psi_{ij} + \theta_i = \arctan\frac{\Delta y_{ij}}{\Delta x_{ij}}.$$

Therefore
$$\dot{\psi}_{ij} + \dot{\theta}_i = \frac{\dot{\Delta}y_{ij}.\Delta x_{ij} - \dot{\Delta}x_{ij}.\Delta y_{ij}}{(\Delta x_{ij})^2 + (\Delta y_{ij})^2},$$

or
$$\dot{\psi}_{ij} + \omega_i = \frac{\dot{\Delta}y_{ij}.\Delta x_{ij} - \dot{\Delta}x_{ij}.\Delta y_{ij}}{l_{ij}^2}. \tag{3.11}$$

By substituting $\Delta x_{ij}, \Delta y_{ij}$ from (3.6), $\dot{\Delta}x_{ij}, \dot{\Delta}y_{ij}$ from (3.9) into (3.11) and taking some simple manipulations, one can obtain the kinematic equation of the angular between two robots as follows.

$$\dot{\psi}_{ij} = \frac{1}{l_{ij}}(v_i \sin\psi_{ij} - v_j \sin\gamma_{ij} + D\omega_j \cos\gamma_{ij} - l_{ij}\omega_i). \tag{3.12}$$

In summary, in this system the kinematic equations for robot i (the leader) are given by equations (3.1). The state of robot j (the follower) is given by $[l_{ij}, \psi_{ij}, \theta_j]^T$ with kinematic equations:

$$\begin{aligned}
\dot{l}_{ij} &= v_j \cos\gamma_{ij} - v_i \cos\psi_{ij} + D\omega_j \sin\gamma_{ij}, \\
\dot{\psi}_{ij} &= \frac{1}{l_{ij}}(v_i \sin\psi_{ij} - v_j \sin\gamma_{ij} + D\omega_j \cos\gamma_{ij} - l_{ij}\omega_i), \\
\dot{\theta}_j &= \omega_j.
\end{aligned} \tag{3.13}$$

Control law

Using standard techniques of I/O linearization [Asada and Slotine, 1986], a control law that gives exponentially convergent solutions in the internal shape variables l_{ij} and ψ_{ij} is given by

$$\begin{aligned} \omega_j &= \frac{\cos\gamma_{ij}}{D}\left(\alpha_2 l_{ij}\left(\psi_{ij}^d - \psi_{ij}\right) - v_i \sin\psi_{ij} + l_{ij}\omega_i + \rho_{ij}\sin\gamma_{ij}\right), \\ v_j &= \rho_{ij} - D\omega_j \tan\gamma_{ij}, \end{aligned} \tag{3.14}$$

where

$$\rho_{ij} = \frac{\alpha_1\left(l_{ij}^d - l_{ij}\right) + v_i\cos\psi_{ij}}{\cos\gamma_{ij}}, \tag{3.15}$$

l_{ij}^d, ψ_{ij}^d respectively are desired length and desired relative angle between two robots, α_1, α_2 are positive constants.

The above control law leads to the following dynamics in the $l-\psi$ variables:

$$\begin{aligned} \dot{l}_{ij} &= \alpha_1(l_{ij}^d - l_{ij}), \\ \dot{\psi}_{ij} &= \alpha_2(\psi_{ij}^d - \psi_{ij}). \end{aligned} \tag{3.16}$$

It is proven in [Desai *et al.*, 1998] that for the motion of the leader robot following a circular path: $v_i = K_1$, $\omega_i = K_2$, θ_j is locally asymptotically convergent to $\theta_i(t) + \psi_{ij}^d - \beta_2 - \arccos\left(K_2/\beta_1\right)$, where

$$\beta_1 = \sqrt{\left(\frac{K_2 l_{ij} + K_1\sin\psi_{ij}^d}{D}\right)^2 + \left(\frac{K_1\cos\psi_{ij}^d}{D}\right)^2}, \quad \beta_2 = \arctan\left(\frac{K_1\cos\psi_{ij}^d}{K_2 l_{ij} + K_1\sin\psi_{ij}^d}\right).$$

Consequently, if the leader robot follows a straight line ($\omega_i = K_2 = 0$), θ_j converges exponentially to the initial orientation of the leader at the beginning of the motion $\theta_i(0)$ (for the leader it is maintained as a constant throughout the motion since $\omega_i = 0 \Rightarrow \theta_i(t) = \theta_i(0)$).

Remarks

1. If $\cos\gamma_{ij} = 0$, the control law is given by

$$\omega_j = \frac{\alpha_1(l_{ij}^d - l_{ij}) + v_i \cos\psi_{ij}}{D\sin\gamma_{ij}},$$

$$v_j = \frac{v_i \sin\psi_{ij} - l_{ij}\omega_i - \alpha_2 l_{ij}(\psi_{ij}^d - \psi_{ij})}{\sin\gamma_{ij}}.$$

2. The optimal point-to-point paths of a nonholonomic car with constraints on the turning radius were shown to be composed of straight lines and circular arcs [Reeds and Shepp, 1990]. The convergence of above solutions has been obtained without placing any restrictions on translational velocity K_1 and angular velocity K_2 of the leader robot. Thus, the $l-\psi$ controlled robot can track, with exponential convergence, any straight line, circular arc, or rotate in place motion of the leader robot thereby following the optimal trajectory generated by the leader.

3.3.2 Separation-bearing control

Adopted by the original l-ψ control, *separation-bearing* controller proposed in [Fierro *et al.*, 2001] could be also used for robot j to follow robot i with a desired separation l_{ij}^d and desired relative bearing ψ_{ij}^d. The control velocities for the follower are given by

$$\begin{aligned} v_j &= s_{ij}\cos\gamma_{ij} - l_{ij}\sin\gamma_{ij}(b_{ij}+\omega_{ij}) + v_i\cos(\theta_i - \theta_j), \\ \omega_j &= \frac{1}{D}\left[s_{ij}\sin\gamma_{ij} + l_{ij}\cos\gamma_{ij}(b_{ij}+\omega_{ij}) + v_i\sin(\theta_i - \theta_j)\right], \end{aligned} \tag{3.17}$$

where D is the distance from the wheel axis to a reference point on the robot (it can be a castor or another point), and

$$\begin{aligned} s_{ij} &= \alpha_1(l_{ij}^d - l_{ij}), \\ b_{ij} &= \alpha_2(\psi_{ij}^d - \psi_{ij}). \end{aligned} \tag{3.18}$$

The closed-loop linearised system

$$\begin{aligned} \dot{l}_{ij} &= \alpha_1 (l_{ij}^d - l_{ij}), \\ \dot{\psi}_{ij} &= \alpha_2 (\psi_{ij}^d - \psi_{ij}), \\ \dot{\theta}_j &= \omega_j, \end{aligned} \tag{3.19}$$

is proven to be stable , that is relative distance and bearing reach their desired values asymptotically, and the internal dynamics of robot j are stable [Fierro *et al.*, 2002].

3.3.3 Virtual robot tracking control

This approach offers a tracking control with arbitrary clearance between two robots [Jongusuk and Mita, 2001]. The model of this control is described in Figure 3.3.

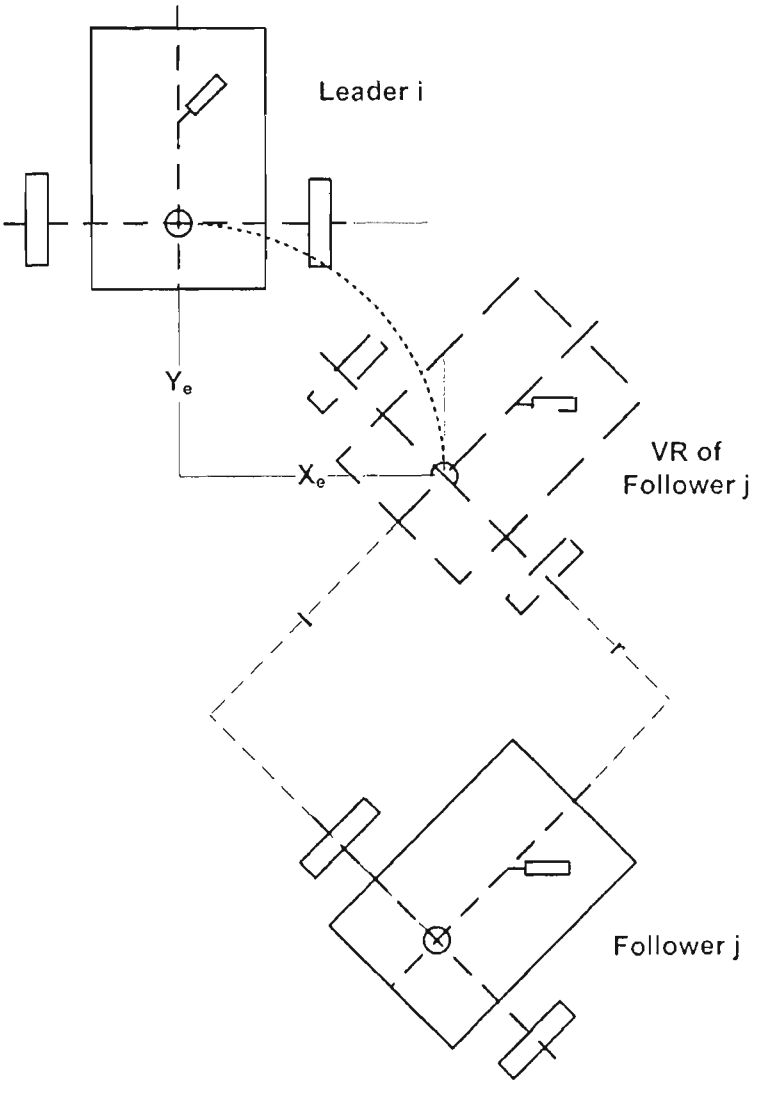


Figure 3.3 Virtual robot tracking model

System model

In order to separate follower from leader robot avoiding collision, a concept of *Virtual Robot* (VR) is defined with additional objective to force error parameters to zero when system approaches final time.

Definition 3.1 *Virtual robot (VR) is a hypothetical robot, whose orientation is identical to that of its corresponding follower robot, but position is placed apart from by the predefined $r-l$ clearances. The symbols l, r denote longitudinal clearance and clearance along rear wheel axis, respectively.*

Let us use some notations: $[x_i, y_i, \theta_i]^T$ refers to leader i, $[x_j, y_j, \theta_j]^T$ refers to follower j, and $[x_{vj}, y_{vj}, \theta_{vj}]^T$ is for VR of follower j.

The relation between VR and its host (i.e. follower j) in terms of positions and orientation is as follows.

$$\begin{aligned} x_{vj} &= x_j - r\sin\theta_j + l\cos\theta_j, \\ y_{vj} &= y_j + r\cos\theta_j + l\sin\theta_j, \\ \theta_{vj} &= \theta_j. \end{aligned} \tag{3.20}$$

From (3.20), we can derive the kinematic model of VR:

$$\begin{bmatrix} \dot{x}_{vj} \\ \dot{y}_{vj} \\ \dot{\theta}_{vj} \end{bmatrix} = \begin{bmatrix} \cos\theta_j & -r\cos\theta_j - l\sin\theta_j \\ \sin\theta_j & -r\sin\theta_j + l\cos\theta_j \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_j \\ \omega_j \end{bmatrix} = \begin{bmatrix} B_{vj} \\ 0 \quad 1 \end{bmatrix} u_j, \tag{3.21}$$

where

$B_{vj} = \begin{bmatrix} \cos\theta_j & -r\cos\theta_j - l\sin\theta_j \\ \sin\theta_j & -r\sin\theta_j + l\cos\theta_j \end{bmatrix}$, and $u_j = \begin{bmatrix} v_j \\ \omega_j \end{bmatrix}$ denotes velocity vector of follower j.

A control law for robot j is designed such that its virtual robot can track leader i with the position errors decreasing exponentially. The error model can be derived as

$$\dot{E}_{vji} = \begin{bmatrix} \dot{x}_{vj} - \dot{x}_i \\ \dot{y}_{vj} - \dot{y}_i \end{bmatrix} = B_{vj} u_j - b_i v_i , \tag{3.22}$$

where v_i is the translational velocity of leader i, and $b_i = \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix}$.

Control law

A standard I/O linearization technique is used to generate the control law:

$$u_j = B_{vj}^{-1}(b_i v_i - \Lambda E_{vji}), \tag{3.23}$$

where $\Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$ is a positive –definite diagonal matrix.

The solution of this controlled system is

$$\begin{aligned} e_x &= x_{vj} - x_i = e_x(0).e^{-\lambda_1 t}, \\ e_y &= y_{vj} - y_i = e_y(0).e^{-\lambda_2 t}, \end{aligned} \tag{3.24}$$

which demonstrate exponential convergence to zero.

This control law ensures a desired $r-l$ configuration of leader i and follower j as $t \to \infty$. Zero dynamics given in term of θ_j are also proven to be stable to guarantee the use of control law (3.23) [Jongusuk and Mita, 2001]. However, there are some limitations of this approach.

Limitations

1. Difficulty in forming a line configuration

l is designed not to be zero in order to matrix B_{vj} is non-singular. This implies that the real robot cannot be parallel to its VR. Therefore, the line configuration between two robots cannot be obtained directly by using this approach.

2. Failure to form some desired configuration without inter-robot collision.

The zero dynamics given in term of θ_j are proved to be stable, but the convergent value of θ_j is not analysed. In some cases the orientation of the VR (and it host) cannot converge to that of the leader. We will show some of these cases.

Applying the proposed control law, one has

$$\omega_j = \dot{\theta}_j = \left(\frac{\lambda_1 e_x - v_i \cos\theta_i}{l}\right)\sin\theta_j + \left(\frac{-\lambda_2 e_y + v_i \sin\theta_i}{l}\right)\cos\theta_j . \qquad (3.25)$$

Consider simple cases when robot i runs on a linear trajectory, i.e. $\omega_i = 0$. Let us assume, without loss of generality, that $\theta_i(0) = 0$. Therefore: $\theta_i(t) = 0$.

Equation (3.25) becomes

$$\dot{\theta}_j = \left(\frac{\lambda_1 e_x - v_i}{l}\right)\sin\theta_j + \left(\frac{-\lambda_2 e_y}{l}\right)\cos\theta_j , \qquad (3.26)$$

or

$$\dot{\theta}_j = K\sin(\theta_j + \varphi) , \qquad (3.27)$$

with $$K = \frac{1}{|l|}\sqrt{(\lambda_1 e_x - v_i)^2 + (-\lambda_2 e_y)^2} ,$$

and $$\varphi = \begin{cases} \arctan\left(\dfrac{-\lambda_2 e_y}{\lambda_1 e_x - v_i}\right) & \text{if } \left(\dfrac{\lambda_1 e_x - v_i}{l}\right) > 0 \\ \arctan\left(\dfrac{-\lambda_2 e_y}{\lambda_1 e_x - v_i}\right) + \pi & \text{if } \left(\dfrac{\lambda_1 e_x - v_i}{l}\right) < 0. \end{cases}$$

Remember here that $e_x = e_x(0).e^{-\lambda_1 t}$ and $e_y = e_y(0).e^{-\lambda_2 t}$. Hence, when t large enough, $|\lambda_1 e_x|$ and $|\lambda_2 e_y|$ are small enough compared to $|v_i|$, we can estimate

$$K \approx \left| \frac{v_i}{l} \right|,$$

and

$$\varphi \approx \begin{cases} 0 & \text{if } \frac{v_i}{l} < 0 \\ \pi & \text{if } \frac{v_i}{l} > 0. \end{cases}$$

Hence,

$$\dot{\theta}_j \approx \begin{cases} K \sin\theta_j & \text{if } \frac{v_i}{l} < 0 \\ -K \sin\theta_j & \text{if } \frac{v_i}{l} > 0. \end{cases} \tag{3.28}$$

In general cases, since $K > 0$, equation (3.28) shows that

$$\theta_i \to \begin{cases} \pi & \text{if } \frac{v_i}{l} < 0 \\ 0 & \text{if } \frac{v_i}{l} > 0. \end{cases} \tag{3.29}$$

For examples, Figure 3.4 and Figure 3.5 depict the orientation of the follower in two different cases of sign of $(\frac{v_i}{l})$.

For both cases, we use Matlab to solve (3.25) with the following parameters:

$$v_i = 5, \omega_i = 0, \theta_i(0) = 0, e_x(0) = 20, e_y(0) = 30, \lambda_1 = 1, \lambda_2 = 2, t = [0, 60], \theta_j(0) = [0, 2\pi].$$

The clearance l used for Figure 3.4 is $l = 10$ and for Figure 3.5 is $l = -10$.

As the above analysis, in Figure 3.4 with $\frac{v_i}{l} = 0.5, \theta_j \to 0\,(or\, 2\pi)$, while in Figure 3.5 with $\frac{v_i}{l} = -0.5$, $\theta_j \to \pi\,(or\, 3\pi)$.

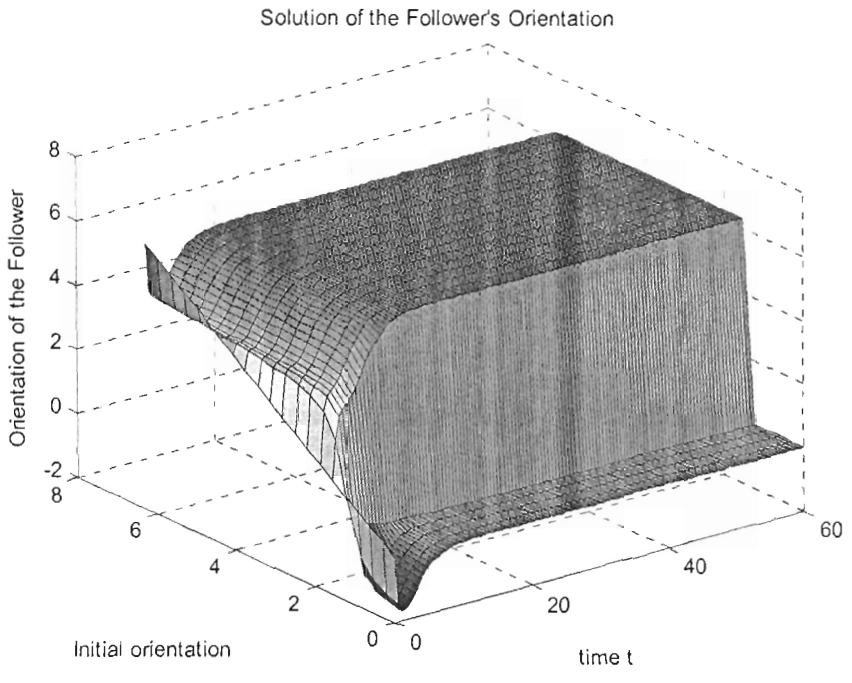


Figure 3.4: Orientation of follower *j* in the case $\frac{v_i}{l} = 0.5 > 0$

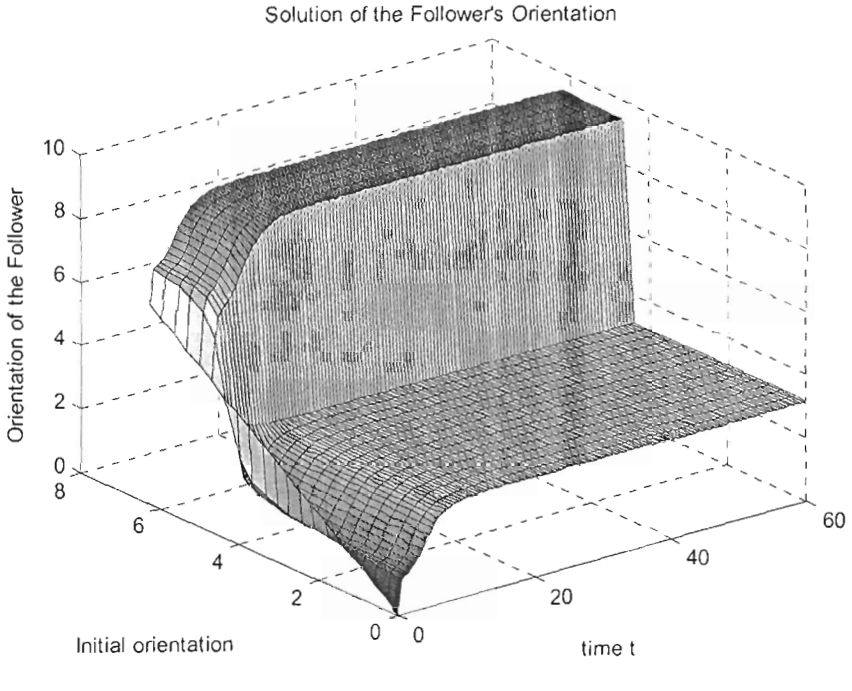


Figure 3.5: Orientation of follower *j* in the case $\frac{v_i}{l} = -0.5 < 0$

Moreover, in the special case $e_x(0) = e_y(0) = 0$, and $\sin\theta_j(0) = 0$ (i.e. $\theta_j(0) = 0$ or $\theta_j(0) = \pi$), if the leader has $\omega_i = 0$, $\theta_i(0) = 0$ (i.e. $\theta_i(t) = 0$), from (3.25) one has $\omega_j = \dot{\theta}_j = 0$. It is clearly that the orientation of the follower (and of its VR) do not always equal to $\theta_i(t) = 0$. For examples: if $\theta_j(0) = 0$ then $\theta_j(t) = 0$, but if $\theta_j(0) = \pi$ then $\theta_j(t) = \pi$.

When the orientation of VR (and its host) converges to a value different from that of the leader, the desired $r-l$ configuration of leader i and follower j cannot be obtained (e.g. see Figure 3.6) and collisions between them may happen.

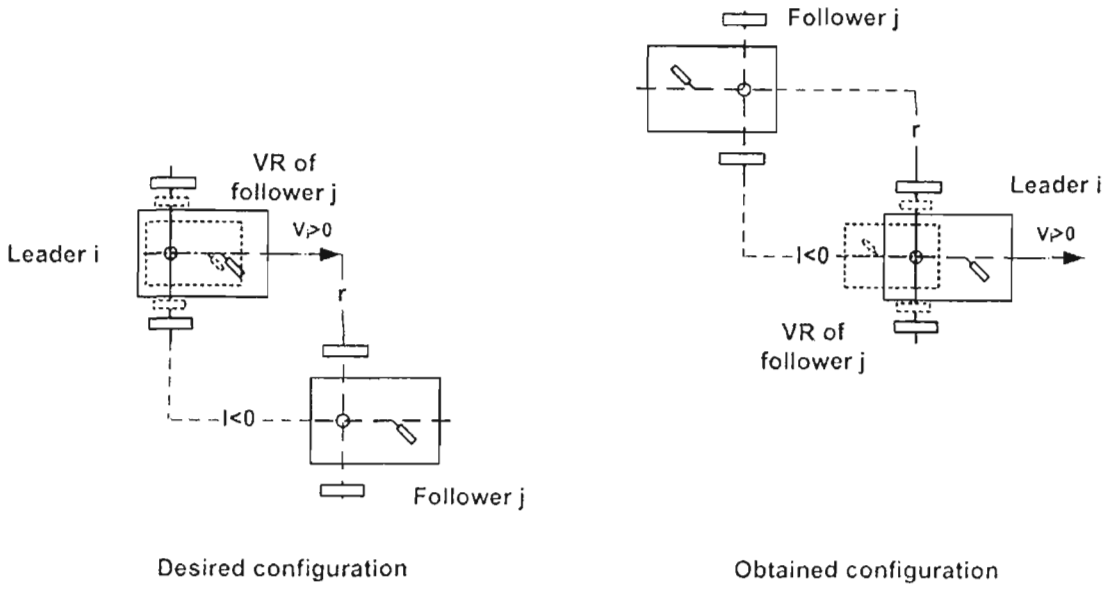


Figure 3.6: An example of impossibility to obtain a desired configuration.

3.4 Virtual head robot tracking control

As above analysis, the *virtual robot tracking* control has some limitations including the impossibility to establish a desired configuration (including line configuration) in some cases and possibility of collision between robots. The following *Virtual Head Robot Tracking* (VHRT) model, motivated by the ideas of virtual robot in [Jongusuk and Mita, 2001] and virtual reference point (x_L, y_L) in [Gustavi and Hu, 2005] is proposed to remedy these circumstances.

3.4.1 System model

Let us firstly redefine the concept of VR as follows [Nguyen *et al.*, 2006b].

Definition 3.2 *Virtual robot (VR) is a hypothetical robot whose orientation is identical to that of its host robot, but position is placed apart from the predefined R-L clearances. The symbols L and R denote respectively the longitudinal clearance and clearance along rear wheel axis.*

The relation between VR and its host in terms of positions and orientation can be written as

$$\begin{cases} x_{vi} = x_i + R\sin\theta_i - L\cos\theta_i \\ y_{vi} = y_i - R\cos\theta_i - L\sin\theta_i \\ \theta_{vi} = \theta_i, \end{cases} \tag{3.30}$$

where (x_i, y_i, θ_i) and $(x_{vi}, y_{vi}, \theta_{vi})$ are respectively coordinates of the host (robot i itself) and its virtual robot. It is noted that R and L here are defined to be strictly positive for the case VR is located in the right-bottom corner of its host, as shown in Figure 3.7. Figure 3.8 describes signs of R, L regarding the relative position of the VR with respect to its host.

Definition 3.3 *Head robot (HR) is a hypothetical robot whose orientation is identical to that of its host, but position is placed at distance $d > 0$ ahead from its host.*

The relation between HR and its host can be written as

$$\begin{cases} x_{hj} = x_j + d\cos\theta_j \\ y_{hj} = y_j + d\sin\theta_j \\ \theta_{hj} = \theta_j, \end{cases} \tag{3.31}$$

where (x_j, y_j, θ_j) and $(x_{hj}, y_{hj}, \theta_{hj})$ denote coordinates of the host (robot j) and its head robot, respectively. Note that HR is designated to serve as a virtual robot of the follower with d to be chosen adequately small as a tracking margin. HR is identical with its host when $d = 0$.

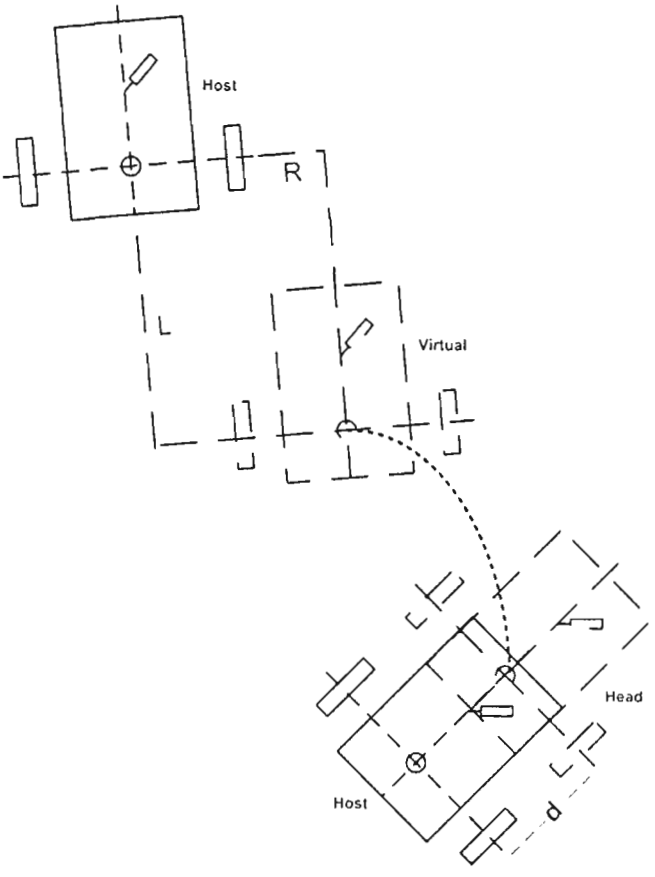


Figure 3.7: VR-HR tracking model

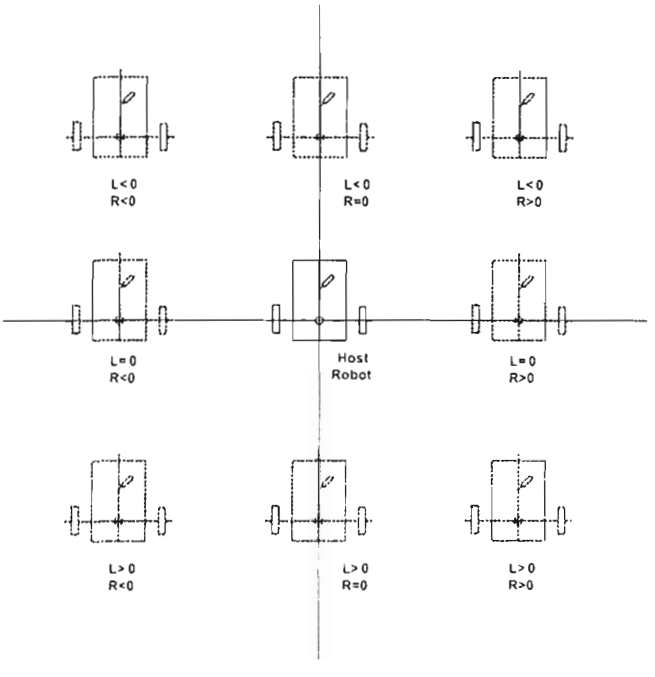


Figure 3.8: Signs of R, L regarding the relative position of the VR with respect to its host

Position errors between VR of leader i and HR of follower j are

$$\begin{cases} e_{xji} = x_{hj} - x_{vi} = (x_j + d\cos\theta_j) - (x_i + R\sin\theta_i - L\cos\theta_i) \\ e_{yji} = y_{hj} - y_{vi} = (y_j + d\sin\theta_j) - (y_i - R\cos\theta_i - L\sin\theta_i). \end{cases} \tag{3.32}$$

A control law for follower robot j is designed such that its head robot can track the virtual robot of leader i with the position errors decreasing monotonically. The control law should ensure a desired R-L configuration of leader i and follower j with position errors smaller or equal the chosen margin d, as $t \to \infty$.

From (3.1) and (3.32) an error model can be derived as

$$\dot{E}_{ji} = B_j u_j - b_i u_i, \tag{3.33}$$

where

$$\begin{cases} E_{ji} = \begin{bmatrix} e_{xji} \\ e_{yji} \end{bmatrix} \\ B_j = \begin{bmatrix} \cos\theta_j & -d\sin\theta_j \\ \sin\theta_j & d\cos\theta_j \end{bmatrix}, u_j = \begin{bmatrix} v_j \\ \omega_j \end{bmatrix} \\ b_i = \begin{bmatrix} \cos\theta_i & R\cos\theta_i + L\sin\theta_i \\ \sin\theta_i & R\sin\theta_i - L\cos\theta_i \end{bmatrix}, u_i = \begin{bmatrix} v_i \\ \omega_i \end{bmatrix}. \end{cases} \tag{3.34}$$

3.4.2 Control law

A standard I/O linearization technique is used to generate the control law:

$$u_j = B_j^{-1}\left[b_i u_i - \Lambda_j E_{ji}\right], \tag{3.35}$$

where $\Lambda_j = \begin{bmatrix} \lambda_{j1} & 0 \\ 0 & \lambda_{j2} \end{bmatrix}$ is a positive–definite diagonal matrix and $B_j^{-1} = \frac{1}{d}\begin{bmatrix} d\cos\theta_j & d\sin\theta_j \\ -\sin\theta_j & \cos\theta_j \end{bmatrix}$.

Time responses of the controlled system errors are then

$$\begin{cases} e_{xji} = e_{xji}(0).e^{-\lambda_{j1}t} \\ e_{yji} = e_{yji}(0).e^{-\lambda_{j2}t}, \end{cases} \tag{3.36}$$

which demonstrate exponential convergence to zero. The control u_j can always be defined if $d > 0$ is chosen such that matrix B_j is non-singular.

3.4.3 Zero dynamics

By applying (3.35), the differential equation for θ_j can be obtained as

$$\dot{\theta}_j = d^{-1}[(\lambda_1 e_{xji} - v_i' \cos\theta_i')\sin\theta_j + (-\lambda_2 e_{yji} + v_i' \sin\theta_i')\cos\theta_j], \tag{3.37}$$

where

$$\begin{aligned} v_i' &= \sqrt{{v_i}^2 + (R\omega_i)^2 + (L\omega_i)^2 + 2Rv_i\omega_i}, \\ \theta_i' &= \text{atan2}(X, Y), \\ X &= v_i \sin\theta_i + (R\sin\theta_i - L\cos\theta_i)\omega_i, \\ Y &= v_i \cos\theta_i + (R\cos\theta_i + L\sin\theta_i)\omega_i. \end{aligned} \tag{3.38}$$

Equation (3.37) is then rewritten as

$$\dot{\theta}_j = A_j \cos(\theta_j + \beta_j), \tag{3.39}$$

where $A_j = d^{-1}\sqrt{(\lambda_1 e_{xji} - v_i' \cos\theta_i')^2 + (-\lambda_2 e_{yji} + v_i' \sin\theta_i')^2}$, and

$$\beta_j = \text{atan2}(-\lambda_1 e_{xji} + v_i' \cos\theta_i',\ -\lambda_2 e_{yji} + v_i' \sin\theta_i').$$

By assigning the right hand side of (3.39) to a steady-state angular velocity ω_{ri} and linearising (3.39) along the solution ${\theta_j}^e = -\beta_j + \arccos(\omega_{ri}/A_j)$, one can obtain

$$\delta\dot{\theta}_j = -A_j \sin(\theta_j + \beta_j)|_{\theta_j = {\theta_j}^e}\ \delta\theta_j = -\sqrt{A_j^2 - {\omega_{ri}}^2}\,\delta\theta_j, \tag{3.40}$$

which implies the stability of zero dynamics.

3.4.4 Critical area

The control law ensures there is no collision between two robots i and j if at the beginning the centre of leader i is not found in the "critical area", determined by initial positions of the two robots as illustrated in Figure 3.9, where the safe distance referred to the centre of any robot in the group for avoidance of collision with others is defined as

$$D_{safe} = d + 2r_{safe}. \tag{3.41}$$

It is clear from the expression of B_j^{-1} that the margin d cannot be chosen too small as this will involve very high angular velocity of robot j.

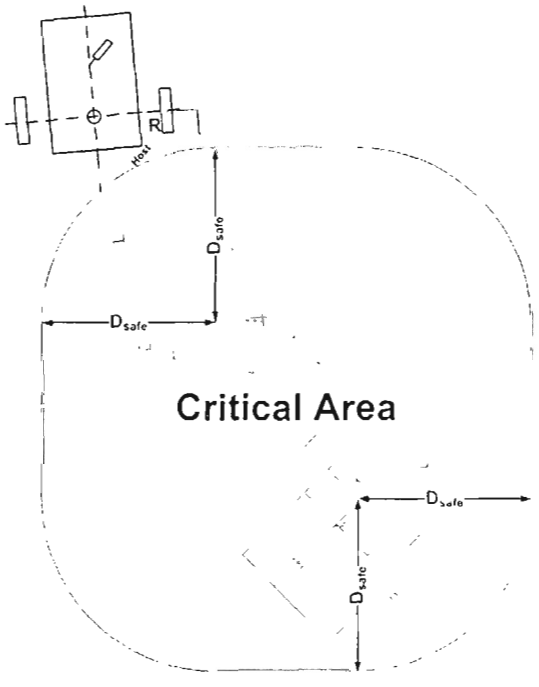


Figure 3.9: Critical area of possible collision

If the condition for collision avoidance is not satisfied, a reactive control scheme will be applied as detailed in the next chapter. In addition, there shall be a parameter constraint which can be described as

$$R^2 + L^2 - d^2 > (2r_{safe})^2, \tag{3.42}$$

which ensures the establishment of the desired configuration between two robots under VHRT control.

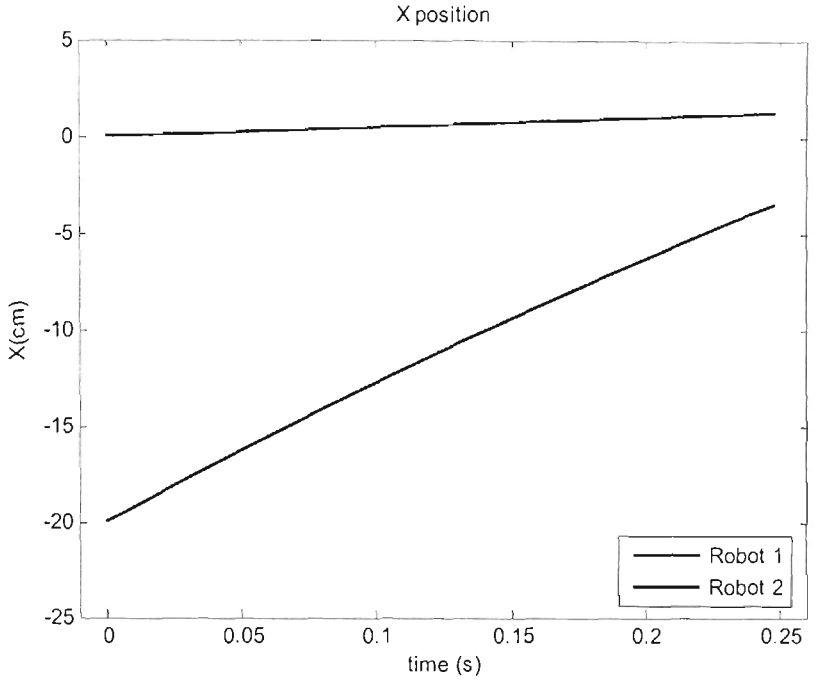
3.5 Simulation results

We present here two simulations to validate the capability of the VHRT control to remedy limitations of VRT control. The first simulation shows that in the same initial conditions of a group of two robots, the VHRT control can be used to obtain a desired leader-follower configuration while applying the VRT control a collision may happen. The second simulation shows a similar case while the VHRT control is useful for obtaining a desired configuration, the group with the VRT control although can avoid collisions but cannot obtain that desired configuration.

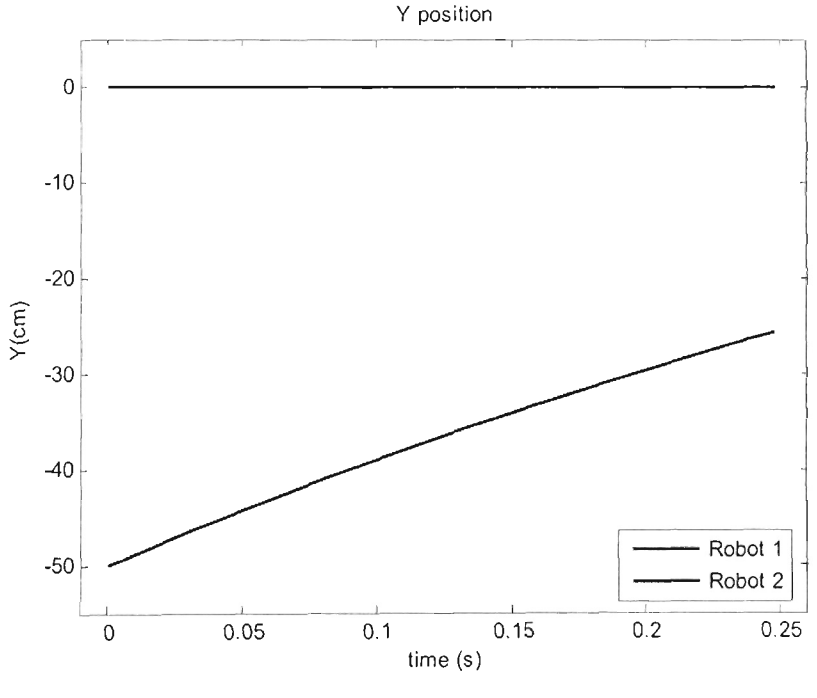
3.5.1 Simulation 3.1: Collision avoidance

To compare the VHRT and VRT control and illustrate the capability of the VHRT to obtain a desired configuration without collisions between robots in a group, let us consider the case of two mobile robots: robot 1 is the leader, robot 2 is the follower. Parameters and conditions used in this simulation were set as in Table 3.1.

Figures 3.10 and 3.11 show the time responses of position X, Y, orientation θ and the trajectories of two mobile robots in the global coordinates with VRT control. Figures 3.12, 3.13 show corresponding results with VHRT control. Considering the safe distance between two robots $2r_{safe} = 26(cm)$, a VRT controller can cause a collision at $t = 0.25s$ and the follower cannot continue to reach its desired position. In this case with $\frac{v_1}{l} = \frac{5}{-40}(s^{-1}) < 0$, as analysed above, robot 2 (the follower) changed its orientation to the opposite one of the leader (robot 1) and followed an unpredictable trajectory in the transient stage. In the same initial conditions and with the similar control parameters, VHRT control can be used to obtain a desired configuration without collisions between robots. The follower (robot 2) when applying VHRT control seems to go directly to its desired position with respect to the leader (robot 1)



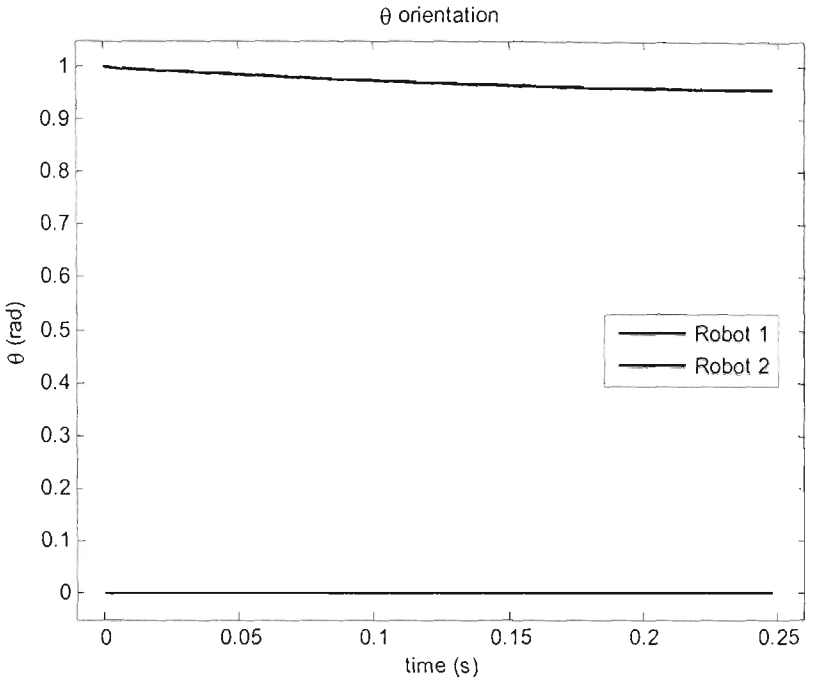
a)



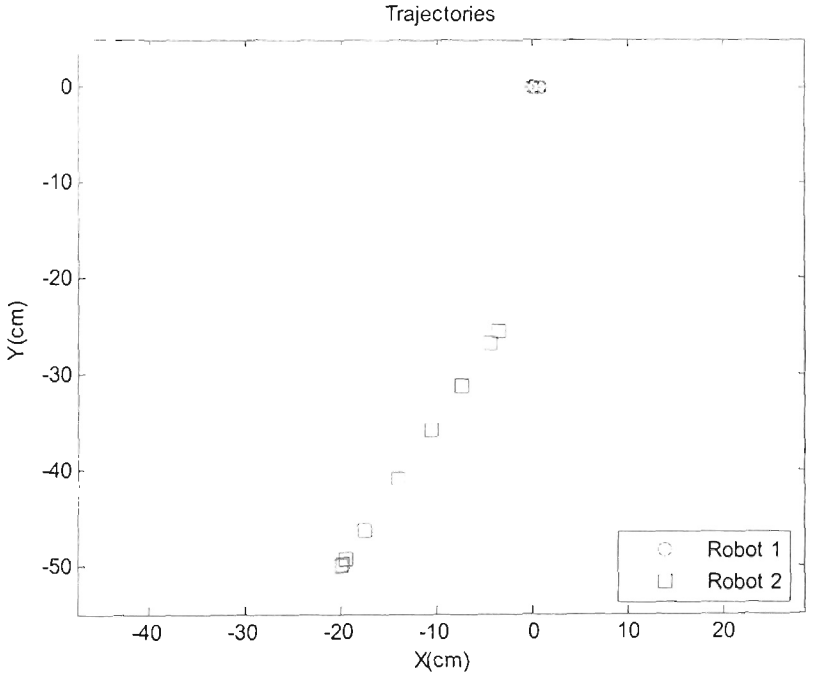
b)

Figure 3.10: Simulation 3.1 results, case of VRT control - X, Y position

a) X position b) Y position

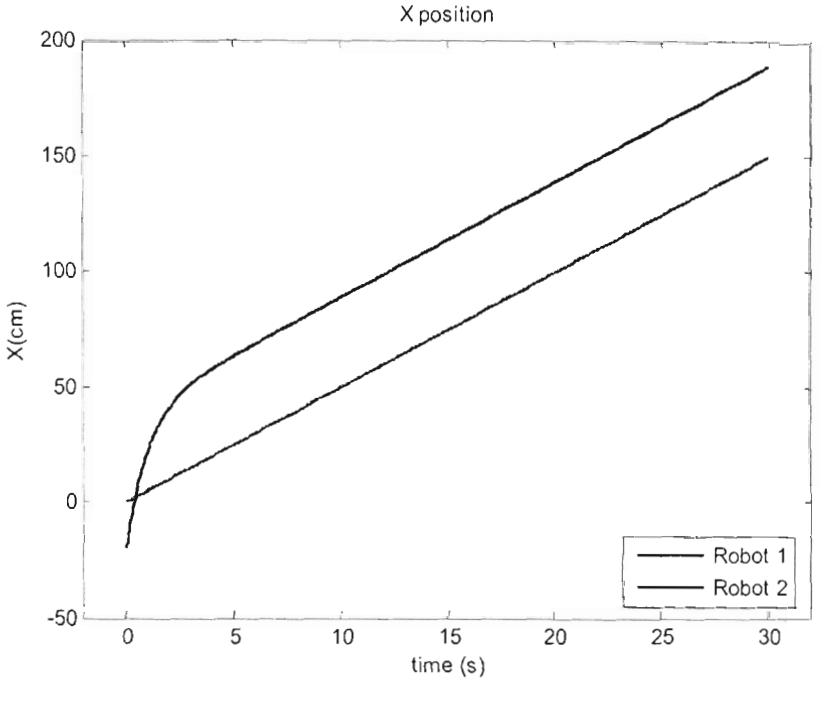


a)

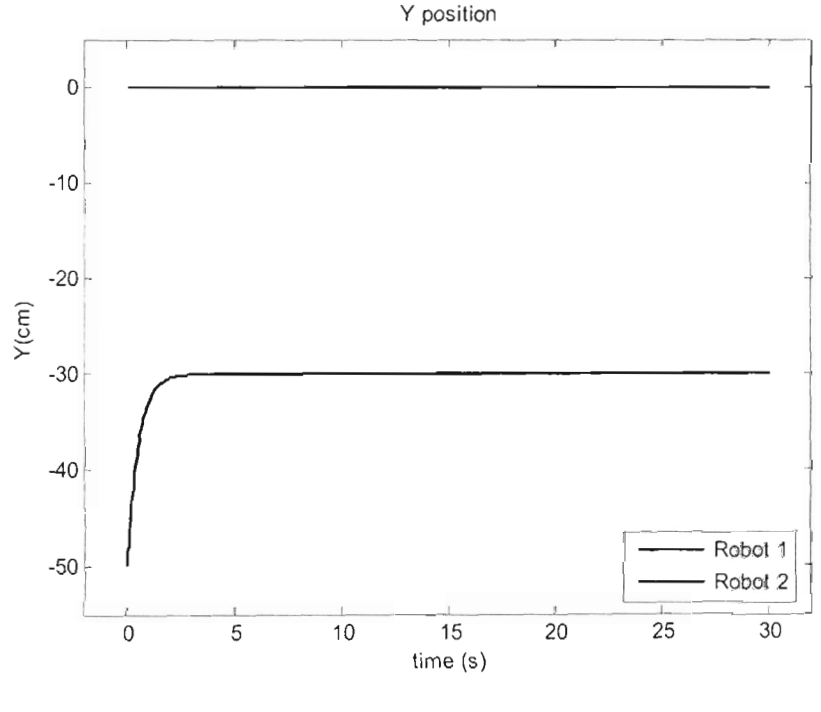


b)

Figure 3.11: Simulation 3.1 results, case of VRT control - θ orientation and trajectories: a) θ- orientation b) Trajectories



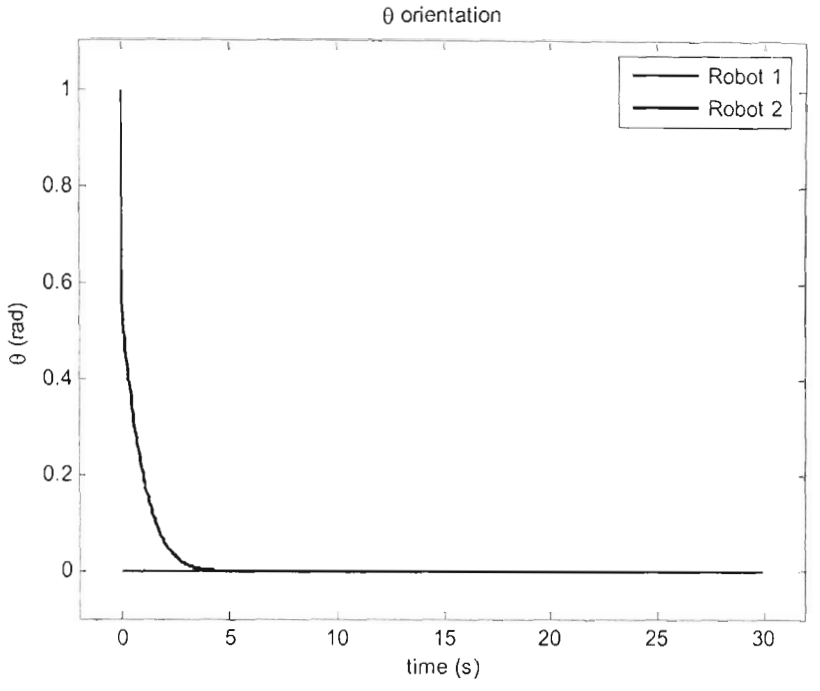
a)



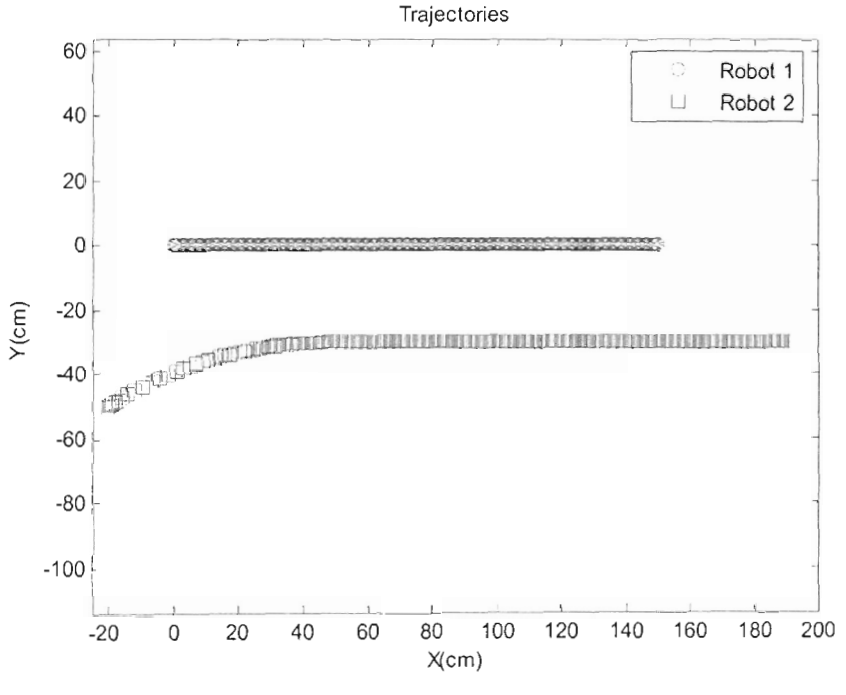
b)

Figure 3.12: Simulation 3.1 results, case of VHRT control - X, Y position

a) X position b) Y position



a)



b)

Figure 3.13: Simulation 3.1 results, case of VHRT control - θ orientation and trajectories: a) θ- orientation b) Trajectories

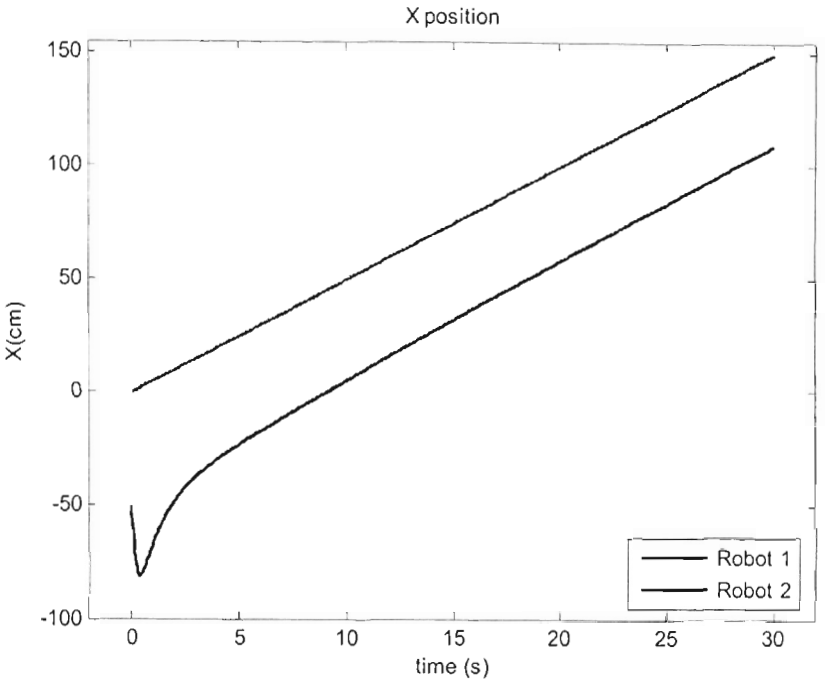
Table 3.1: Parameters and Conditions for Simulation 3.1

Simulation Parameters		Robot 1	Robot 2
Initial Conditions	$x(0)$ (cm)	0	-20
	$y(0)$ (cm)	0	-50
	$\theta(0)$ (rad)	0	1
	v (cm/s)	5	
	ω (rad/s)	0	
Safe distance between any two robots	$2r_{safe}$ (cm)	26	26
Parameters for VR in VRT control	r (cm)		30
	l (cm)		-40
Parameters for VR in VHRT control	R (cm)		30
	L (cm)		-40
Tracking margin for head robot in VHRT control	d (cm)		1
Parameters for VHT control	λ_1 (s^{-1})		1
	λ_2 (s^{-1})		2
Parameters for VHRT control	λ_1 (s^{-1})		1
	λ_2 (s^{-1})		2

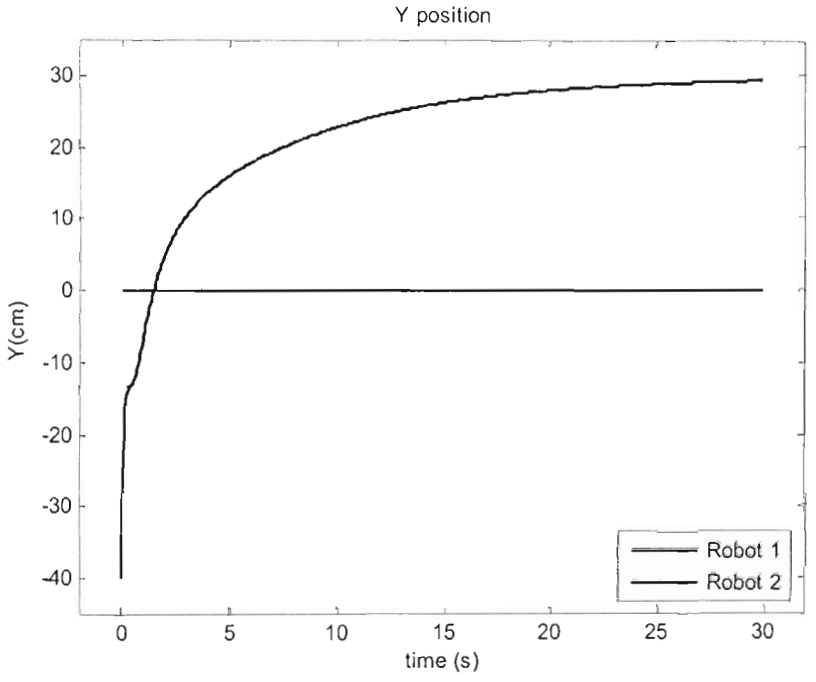
3.5.2 Simulation 3.2: Desired configuration

This simulation demonstrates the case the VRT control cannot be used to obtain a desired configuration while VHRT control can achieve a desired response. Again we consider a group with leader robot 1 and follower robot 2. Parameters and conditions used in this simulation were set as in Table 3.2

Figures 3.14 and 3.15 show the time responses of position X, Y, orientation θ and the trajectories of two mobile robots in the global coordinates with VRT control. Figures 3.16, 3.17 show corresponding results with VHRT control. Using the same parameters and conditions as in Simulation 3.1, except the initial position and orientation of robot 2, it can be seen that VRT control although can avoid collisions between two robots, the desired configuration cannot be obtained. Because the initial position of robot 2 is quite far from robot 1, the change in orientation of robot 2 (the follower) compared to robot 1 (the leader) although could not cause inter-robot collisions, but it make the group impossible to obtain the desired configuration. At the same condition, VHRT control can achieve a desired response with the similar group behaviours as in Simulation 3.1.



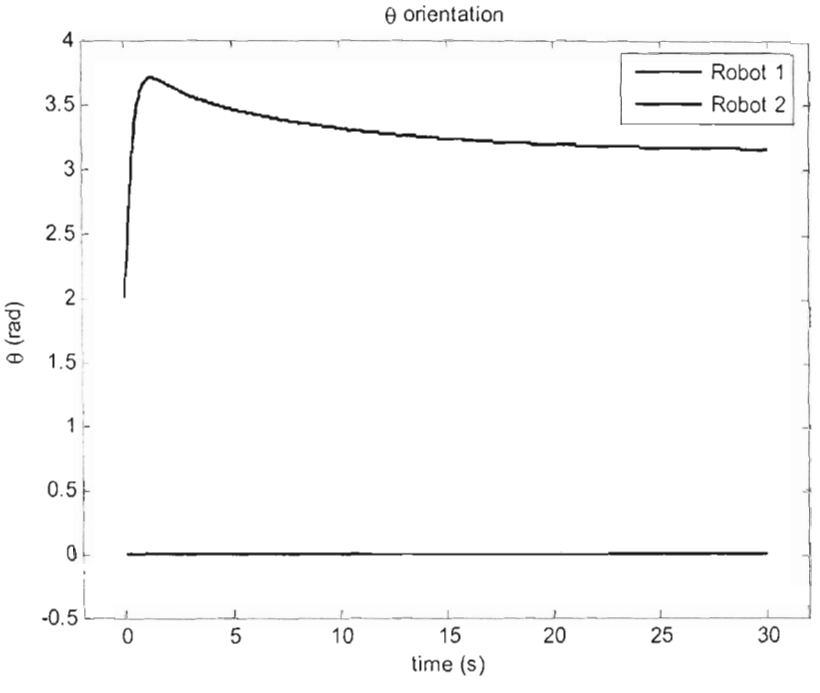
a)



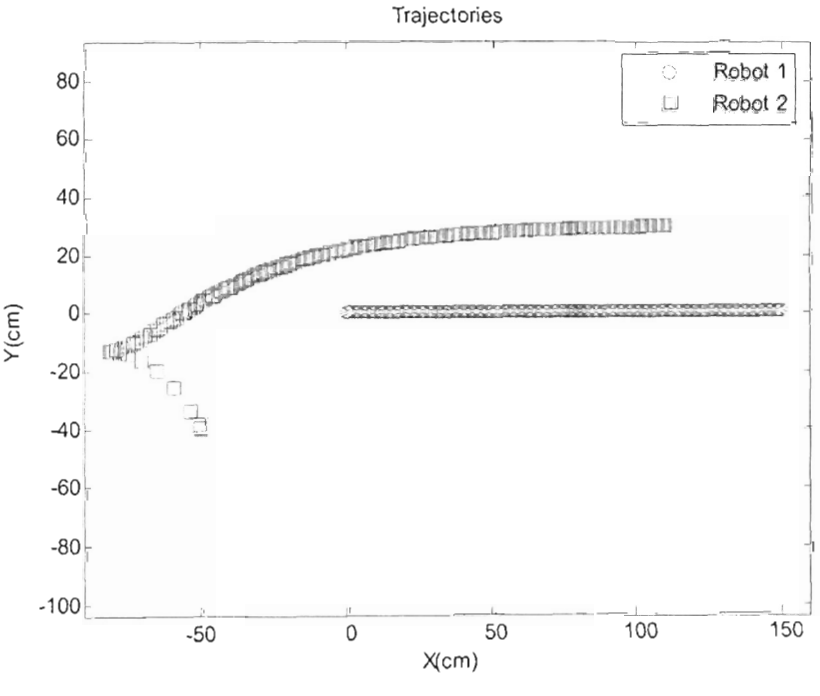
b)

Figure 3.14: Simulation 3.2 results, case of VRT control - X, Y position

a) X position b) Y position

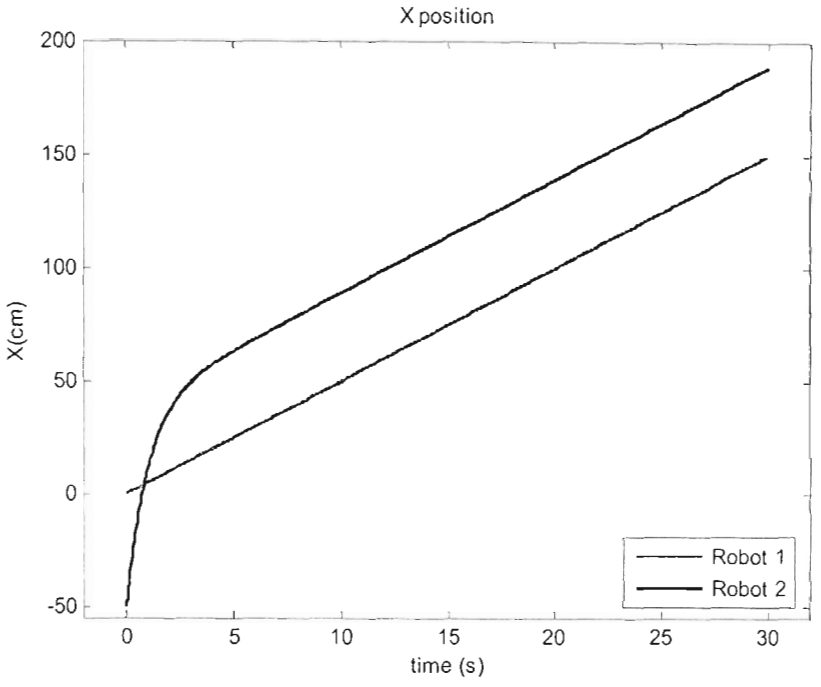


a)

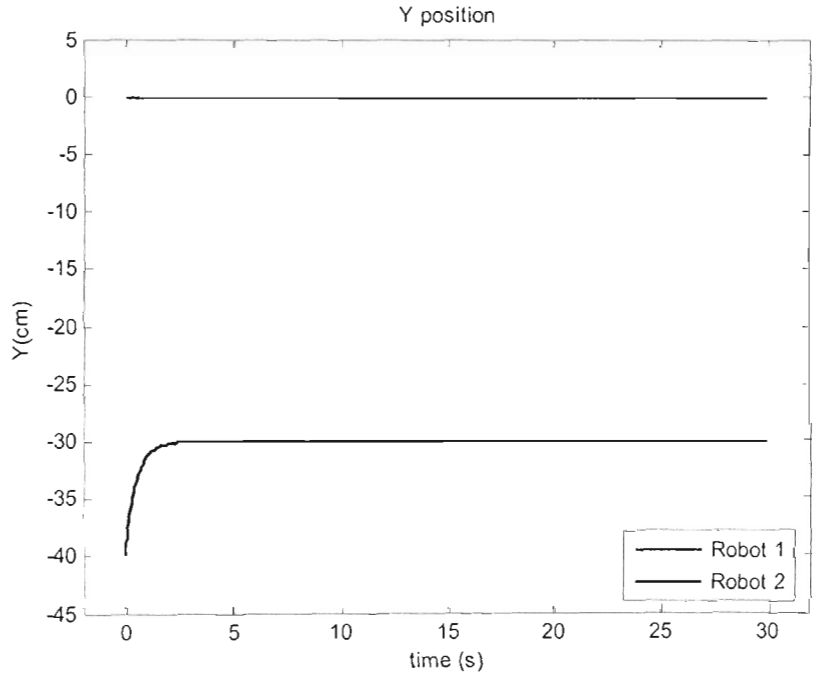


b)

Figure 3.15: Simulation 3.2 results, case of VRT control - θ orientation and trajectories: a) θ- orientation b) Trajectories



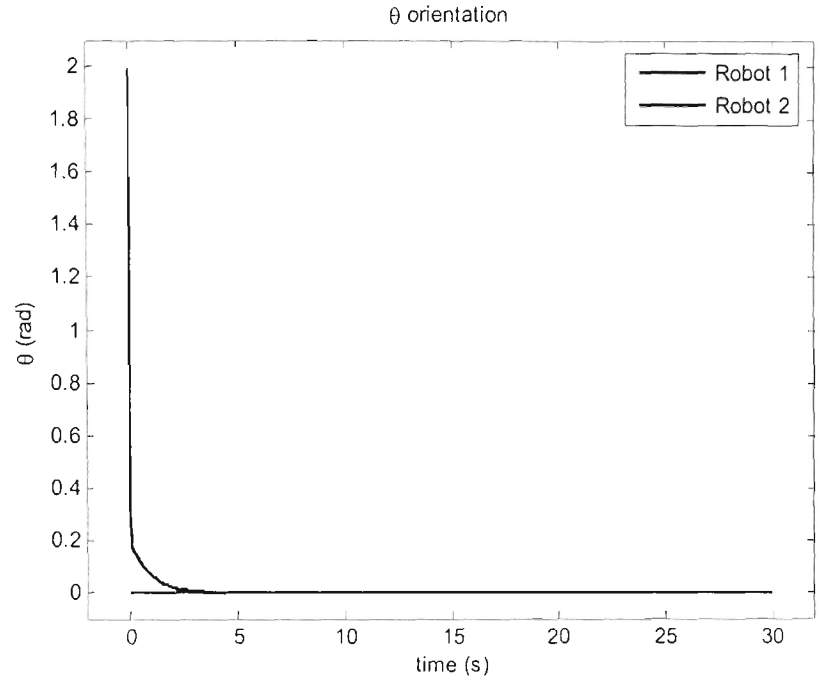
a)



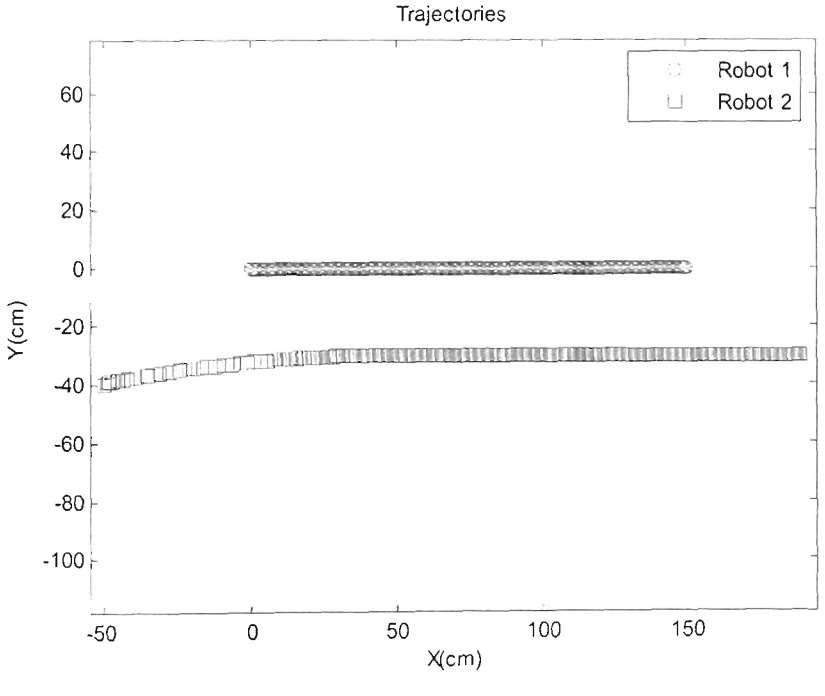
b)

Figure 3.16: Simulation 3.2 results, case of VHRT control - X, Y position

a) X position b) Y position



a)



b)

Figure 3.17: Simulation 3.2 results, case of VHRT control - θ orientation and trajectories: a) θ- orientation b) Trajectories

Table 3.2: Parameters and Conditions for Simulation 3.2

Simulation Parameters		Robot 1	Robot 2
Initial Conditions	$x(0)$ (cm)	0	-50
	$y(0)$ (cm)	0	-40
	$\theta(0)$ (rad)	0	2
	v (cm/s)	5	
	ω (rad/s)	0	
Safe distance between any two robots	$2r_{safe}$ (cm)	26	26
Parameters for VR in VRT control	r (cm)		30
	l (cm)		-40
Parameters for VR in VHRT control	R (cm)		30
	L (cm)		-40
Tracking margin for head robot in VHRT control	d (cm)		1
Parameters for VHT control	λ_1 (s^{-1})		1
	λ_2 (s^{-1})		2
Parameters for VHRT control	λ_1 (s^{-1})		1
	λ_2 (s^{-1})		2

3.6 Conclusion

This chapter has presented a new *virtual head robot tracking* control to establish any desired leader-follower configuration between two mobile robots. This controller can be used as a basic controller in leader-follower strategies to solve the robotic formation problems. Some limitations of *virtual robot tracking* control have been overcome. The contribution of this control law is the capability of forming a desired configuration of two mobile robots from an arbitrary initialisation conditions and maintaining its collision-free shape. A desired formation of a group of multiple mobile robots can be obtained using these basic controllers by letting robots be controlled in appropriate chains of leader–follower relationships.

Using standard I/O feedback linearization technique, the main disadvantage of this controller is that the robustness cannot be guaranteed in the presence of parameter uncertainty or unmodeled dynamics. Moreover, information of position and orientation of two concerning robots is needed for the follower. Accumulated errors in sensor readings (for example getting by encoders) can cause an undesired behaviour of the

follower, especially when initial positions of robots are far from their desired positions. It makes experiments difficult with real mobile robots and requires further navigation assistive algorithms for robot localization.

Using the proposed control approach, collisions between robots may be avoided in most of the circumstances, except cases when the leader is initially located in "critical area" as analysis in Section 3.4.4. A reactive control scheme using 3PLL control may be used to deal with those cases. These control and scheme will be described in the next chapter.

Chapter 4

Three point *l-l* control for formation control

4.1 Introduction

From the previous chapter, it is demonstrated that for a group of three mobile robots, a desired formation can be established by using two *virtual head robot tracking* controllers to obtain an appropriate leader-follower configuration. The problem as to how to avoid collisions among simultaneously three robots in the group or collisions between the follower and leader, which is initially located in "critical area" (Figure 3.9), remains however questionable, and will be the subject of this chapter.

A modified *l-l* control proposed in [Jongusuk and Mita, 2001] could be used for collision avoidance among robots in these circumstances. Notably, this control law, as well as the original *l-l* control [Desai *et al.*, 1998] and the *separation-separation* control [Fierro *et al.*, 2001; Fierro *et al.*, 2002], are subject to a singularity when the reference points of three robots lie on the same line connecting them. This chapter describes a new technique, called *Three Point l-l* (3PLL) control, to deal with this problem. In order to ensure inter-robot collision avoidance, a reactive control scheme is proposed. Finally, an algorithm that combines VHRT and 3PLL control techniques incorporated

with the reactive control scheme is then suggested to initialize and maintain a desired formation for a group of three mobile robots.

Section 4.2 presents the problem formulation. An overview of such control techniques as $l-l$, *separation-separation*, and modified *l-l* control is given in Section 4.3. Details of the 3PLL control design are presented in Section 4.4. Section 4.5 describes the reactive control scheme. Section 4.6 presents the proposed algorithm combining VHRT and 3PLL control for initialisation and establishment of formations for a group of three mobile robots. Simulation results and discussion are presented tin Section 4.7. A conclusion is given in Section 4.8.

4.2 Model and problem formulation

4.2.1 A nonholonomic mobile robot model and collision detection model

The kinematic model is used here as in previous chapter for a three-wheel mobile robot:

$$\begin{aligned} \dot{x} &= v\cos\theta, \\ \dot{y} &= v\sin\theta, \\ \dot{\theta} &= \omega, \end{aligned} \tag{4.1}$$

where $z=(x,y)^T$ is the centre point on the wheel axis, $\theta \in R$ is the orientation and input v,ω is the translational and angular velocities respectively.

The possibility of collision between any two mobile robots i and j is also measured by using function f_{ij} as in (3.2) and (3.3).

4.2.2 Assumptions

Let us introduce some assumptions:

- **Assumption 1**: All robots are of the same model as described in (4.1) and strictly satisfy pure rolling and non-slipping conditions. Function f_{ij} for

assessing the possibility of collision between any two robots i and j is always available.

- **Assumption 2**: Each robot is indexed by a number indicating the priority level according to its role in the group. The lower the index, the higher the priority, with 1 being the leader's index. The follower always follows one or two others which have lower indices.

- **Assumption 3**: Each robot can get any necessary information about its position and orientation and information of its leader in a global coordinates from its communication channel.

- **Assumption 4**: The leader follows a smooth trajectory and the workspace is flat and obstacle-free.

4.2.3 Problem statement

The objective here is to design controllers for robots in a group of three mobile robots to achieve:

1. any desired formation,
2. no collision between any two mobile robots, and
3. group motion satisfying the limitation of communication range.

4.3 Original *l-l* control, separation-separation control, and modified *l-l* control

This section describes the *l-l* based control laws for a group of three mobile robots.

4.3.1 Original *l-l* control

This control law was proposed by Desai *et al.*(1998). Figure 4.1 shows a system of three nonholonomic mobile robots. The aim is to stabilize the distance of the third

robot from the other two robots. The state of the third robot therefore is specified by three independent variables, in this case they are chosen as $[l_{13}, l_{23}, \theta_3]^T$. Distances are measured from the centre of the axle of the first two robots to the castor of the third robot which is offset by D from its axle.

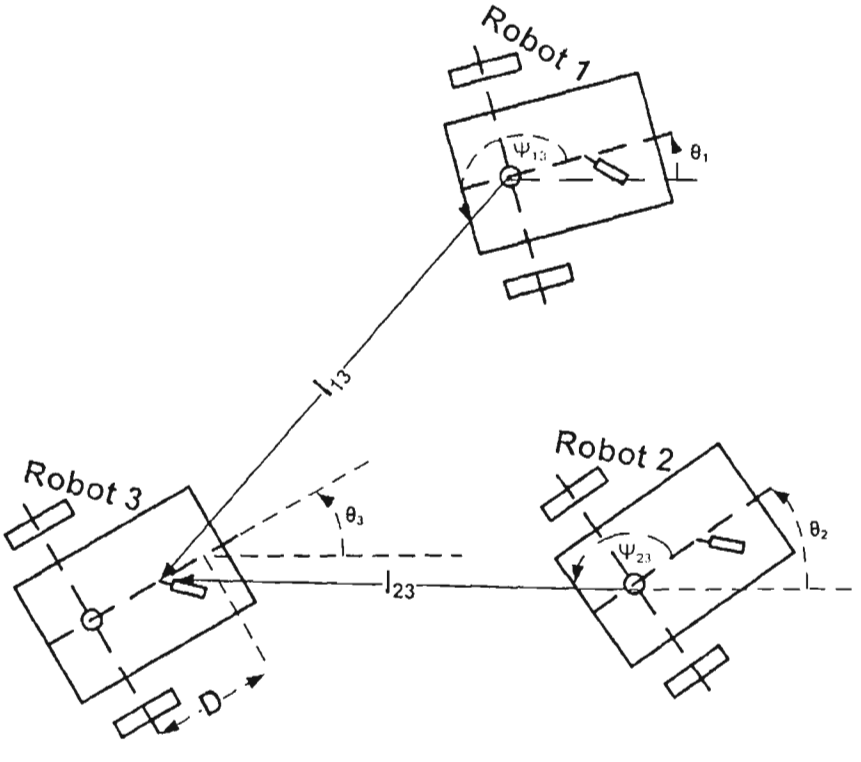


Figure 4.1: Notation for *l-l* control

The goal of the feedback controller is to maintain the desired lengths, l_{13}^d and l_{23}^d of the third robot from its two leaders. The kinematic equations for the system is given by equation (4.1) for robot 1 and robot 2, and

$$\begin{aligned} \dot{l}_{13} &= v_3 \cos\gamma_1 - v_1 \cos\psi_{13} + D\omega_3 \sin\gamma_1, \\ \dot{l}_{23} &= v_3 \cos\gamma_2 - v_2 \cos\psi_{23} + D\omega_3 \sin\gamma_2, \\ \dot{\theta}_3 &= \omega_3, \end{aligned} \tag{4.2}$$

for the third robot; where $\gamma_i = \theta_i + \psi_{i3} - \theta_3 \quad (i = 1,2)$. Note that, first two equations in (4.2) describe relationships in terms of distance between robot 3 and robot 1, robot 2 respectively as in *l-ψ* control (see Section 3.3.1).

The I/O linearization is used to generate a feedback control law:

$$\omega_3 = \frac{\alpha_1(l_{13}^d - l_{13})\cos\gamma_2 + v_1\cos\psi_{13}\cos\gamma_2 - \alpha_2(l_{23}^d - l_{23})\cos\gamma_1 - v_2\cos\psi_{23}\cos\gamma_1}{D\sin(\gamma_1 - \gamma_2)},$$
$$v_3 = \frac{\alpha_1(l_{13}^d - l_{13}) + v_1\cos\psi_{13} - D\omega_3\sin\gamma_1}{\cos\gamma_1}, \tag{4.3}$$

where α_1 and α_2 are positive constants.

This gives exponential convergence for the controller variables:

$$\begin{aligned} \dot{l}_{13} &= \alpha_1(l_{13}^d - l_{13}), \\ \dot{l}_{23} &= \alpha_2(l_{23}^d - l_{23}). \end{aligned} \tag{4.4}$$

Desai *et al.* (1998) also proved that for a straight line parallel motion of the first two robots (i.e. constant velocity $v_1 = v_2 = K, \omega_1 = \omega_2 = 0$ and $\theta_1(0) = \theta_2(0) = \theta_0$), the orientation of robot 3 θ_3 locally converges exponentially to $\theta_3^e = \theta_0$ and $\psi_{13}(t) = \psi_{13}(0)$, $\psi_{23}(t) = \psi_{23}(0)$.

Remarks

1. Equation (4.4) shows that the convergence of the system states is asymptotical but not established within a finite time.

2. The singularity $\sin(\gamma_1 - \gamma_2) = 0$ happens when the reference points of three robots lie on the same line connecting them.

3. If $\cos\gamma_1 = 0$, $\sin(\gamma_1 - \gamma_2) \neq 0$, the control law is given by

$$\omega_3 = \frac{\alpha_1(l_{13}^d - l_{13})\cos\gamma_2 + v_1\cos\psi_{13}\cos\gamma_2 - \alpha_2(l_{23}^d - l_{23})\cos\gamma_1 - v_2\cos\psi_{23}\cos\gamma_1}{D\sin(\gamma_1 - \gamma_2)},$$
$$v_3 = \frac{\alpha_2(l_{23}^d - l_{23}) + v_2\cos\psi_{23} - D\omega_3\sin\gamma_2}{\cos\gamma_2}.$$

4.3.2 Separation-separation control

By adopting the original *l-l* control, Fierro *et al.* (2001) proposed *separation-separation* controllers for robot 3 to follow robot 1 and robot 2 with desired separations l_{13}^d and l_{23}^d, respectively. In this case the control velocities for the follower robot become

$$\begin{aligned} v_3 &= \frac{s_{13}\sin\gamma_2 - s_{23}\sin\gamma_1 + v_1\cos\psi_{13}\sin\gamma_2 - v_2\cos\psi_{23}\sin\gamma_1}{\sin(\gamma_2-\gamma_1)}, \\ \omega_3 &= \frac{-s_{13}\cos\gamma_2 + s_{23}\cos\gamma_1 - v_1\cos\psi_{13}\cos\gamma_2 + v_2\cos\psi_{23}\cos\gamma_1}{D\sin(\gamma_2-\gamma_1)}, \end{aligned} \tag{4.5}$$

where D is the distance from the wheel axis to a reference point on the robot, and

$$\begin{aligned} s_{13} &= \alpha_1(l_{13}^d - l_{13}), \\ s_{23} &= \alpha_2(l_{23}^d - l_{13}). \end{aligned} \tag{4.6}$$

The closed-loop linearised system

$$\begin{aligned} \dot{l}_{i3} &= \alpha_1(l_{13}^d - l_{13}), \\ \dot{l}_{23} &= \alpha_2(l_{23}^d - l_{23}), \\ \dot{\theta}_3 &= \omega_3, \end{aligned} \tag{4.7}$$

is proven to be stable, that is relative distances reach their desired values asymptotically, and the internal dynamics of robot 3 are stable [Fierro *et al.*, 2002].

4.3.3 Modified *l-l* control

This control proposed by Jongusuk and Mita (2001) is used as collision avoidance controller, which requires safe distances between considered robots must be established within a finite time.

Equation (4.2) can be rewritten as

$$\begin{aligned}\begin{bmatrix}\dot{l}_{13}\\ \dot{l}_{23}\end{bmatrix} &= \begin{bmatrix}\cos\gamma_1 & D\sin\gamma_1\\ \cos\gamma_2 & D\sin\gamma_2\end{bmatrix}\begin{bmatrix}v_3\\ \omega_3\end{bmatrix} - \begin{bmatrix}v_1\cos\psi_{13}\\ v_2\cos\psi_{23}\end{bmatrix}\\ &= B_u u_3 - v_u,\\ \dot{\theta}_3 &= \omega_3,\end{aligned} \tag{4.8}$$

where

$$B_u = \begin{bmatrix}\cos\gamma_1 & D\sin\gamma_1\\ \cos\gamma_2 & D\sin\gamma_2\end{bmatrix},\quad u_3 = \begin{bmatrix}v_3\\ \omega_3\end{bmatrix},\quad v_u = \begin{bmatrix}v_1\cos\psi_{13}\\ v_2\cos\psi_{23}\end{bmatrix}.$$

Let us firstly present some mathematical preliminaries on terminal attractor

Terminal attractor

Dynamical systems that come from applications tend to be dissipative: if it were not for some driving force the motion would cease. The dissipation may come from internal friction, thermodynamic losses, or loss of material, among many causes. The dissipation and the driving force tend to combine to kill out initial transients and settle the system into its typical behaviour. The part of the phase space of the dynamical system corresponding to the typical behaviour is the attracting set or attractor [Barhen et al., 1989; Zak, 1989].

At equilibrium, the fixed points of an N-dimensional, dissipative dynamical system

$$\dot{x}_n - f_n(x_1, x_2, \ldots, x_n) = 0, \tag{4.9}$$

for $n = 1,2,\ldots,N$, are defined as its constant solution $x_n(\infty)$. If the real parts of the eigenvalues μ_ξ of the Jacobian matrix $M_{nm} = [\partial f_n / \partial x_m]$ at a fixed point are all negative, that is, $\mathrm{Re}\{\mu_\xi\} < 0$, then these points are locally asymptotically stable. Such points are called regular attractors, since each motion along the phase curve that gets close enough to $\bar{u}(\infty)$, that is, enters a so-called basin of attraction, approaches the corresponding constant value as a limit when t tends to infinity.

Typically, dynamical systems such as (4.9) obey the Lipschitz condition

$$\left|\partial f_n / \partial x_m\right| < \infty, \tag{4.10}$$

which guarantees the existence of a unique solution for each initial condition $\bar{x}(0)$. Theoretically, the system relaxation time to an attractor is infinite, because the transient solution cannot intersect the corresponding solution to which it tends. Figure 4.2a shows the temporal evolution of such an attractor with a simple system: $\dot{x} = -x$ from different initial conditions.

In contrast, the notion of terminal attractor is based on violation of the Lipschitz condition at equilibrium points. As a result of this violation, a fixed point becomes a singular solution enveloping the family of regular solutions, while each regular solution approaches the terminal attractor in finite time.

For example, the following simple one-dimensional system

$$\dot{x} = -x^{1/3}, \tag{4.11}$$

has an attracting equilibrium point at $x = 0$ which violates the Lipschitz condition,

$$\left|d\dot{x} / dx\right| = \left|-1/3x^{-2/3}\right| \to \infty, \quad when\ x \to 0.$$

The attractor is termed terminal, since from any initial condition $x_0 \neq 0$, the dynamical system in (4.11) reaches the equilibrium point $x = 0$ in a finite time,

$$t_0 = -\int_{x_0}^{x \to 0} x^{-1/3} dx = (3/2)x_0^{2/3}.$$

The behavior of this terminal attractor is shown in Figure 4.2b with deferent initial conditions.

A more general case is

$$\dot{x} = -x^k, \quad k > 0,$$

for which the relaxation time for the attractor is

$$t_0 \begin{cases} \to \infty & \text{if } k = i \quad : \text{Regular attractor} \\ = \dfrac{x_0^{1-k}}{1-k} & \text{if } k = \dfrac{1}{2i+1} \quad : \text{Terminal attractor,} \end{cases}$$

where $i \in N^+$ and $N^+ = N \setminus \{0\}$.

We now present some results on terminal attractor.

Proposition 4.1 *The origin of a differential equation*

$$\dot{z} = -sign(z(0))\lambda z^{p/q}, \quad \lambda > 0 \tag{4.12}$$

is a terminal attractor, with a finite reaching time

$$T_r = sign(z(0))\frac{z(0)^{1-\frac{p}{q}}}{\lambda(1-\frac{p}{q})}, \tag{4.13}$$

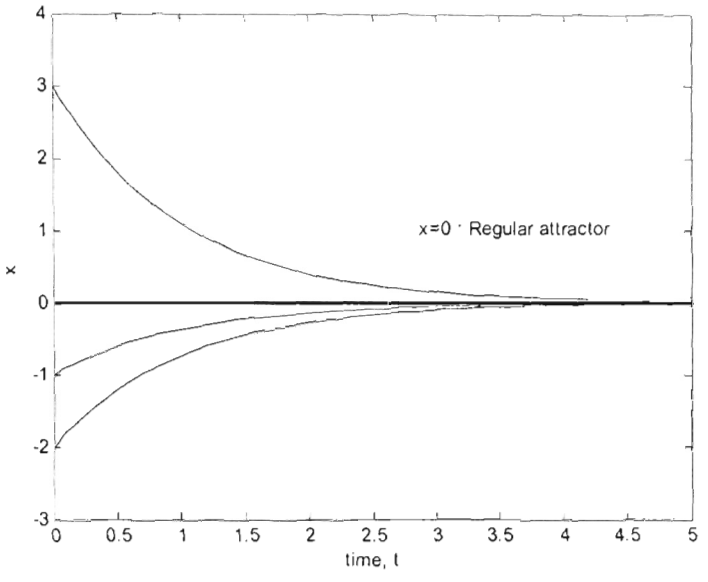
where $(p,q) \in \Omega$ *with*

$$\Omega = \left\{ (p,q) \middle| p = 2m, q = 2n-1, m < n, m,n \in N^+ \right\} \text{ and } N^+ = N \setminus \{0\}$$

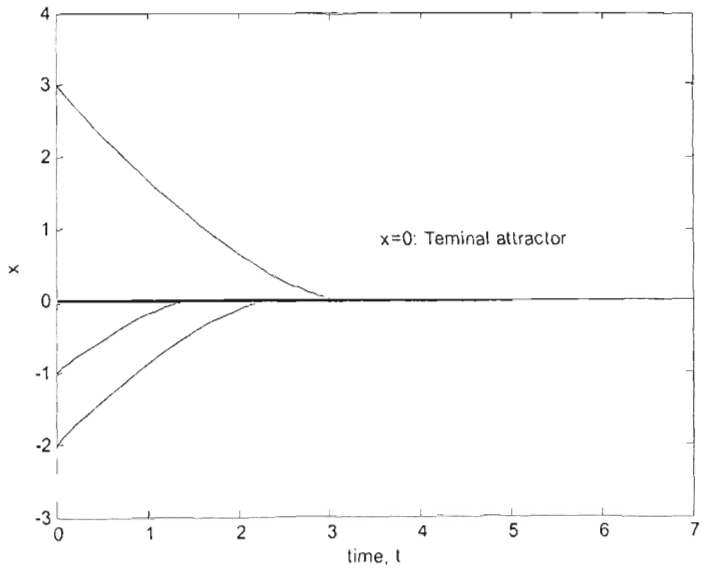
Proof

We consider the initial value of z at $t = 0$.

- If $z(0) < 0$, equation (4.12) shows that $\dot{z} > 0$ if $z \neq 0$. Therefore z increases and reaches to 0.
- If $z(0) > 0$, equation (4.12) shows that $\dot{z} < 0$ if $z \neq 0$. Therefore z decreases and reaches to 0.
- If $z(0) = 0$, the system maintains $z = 0, \forall t$.



a)



b)

Figure 4.2: Temporal evolution of attractors

a) Regular attractor with $\dot{x} = -x$ b) Terminal attractor with $\dot{x} = -x^{1/3}$

Hence $z = 0$ is the attractor of the system (4.12). Moreover, it is easy to check that the Lipschitz condition is violated. Therefore $z = 0$ is a terminal attractor.

Equation (4.12) can be written as

$$sign(z(0))\lambda dt = -z^{-\frac{p}{q}} dz . \tag{4.14}$$

Direct integration of (4.14) from 0 to t yields

$$z(t)^{1-\frac{p}{q}} = z(0)^{1-\frac{p}{q}} - (1-\frac{p}{q})sign(z(0))\lambda t . \tag{4.15}$$

Requiring $z(T_r) = 0$ implies formula (4.13).

■

Now, the terminal attractor can be applied to obtain a modified *l-l* control law.

Required that the controlled variables l_{13} and l_{23} monotonically converge to l_{13}^d and l_{23}^d, respectively, within finite time T_r, we consider z as $l_{i3}^d - l_{i3}, i = 1,2$ and set

$$\begin{bmatrix} \dot{l}_{13}^d - \dot{l}_{13} \\ \dot{l}_{23}^d - \dot{l}_{23} \end{bmatrix} = -\begin{bmatrix} \alpha_1^* & 0 \\ 0 & \alpha_2^* \end{bmatrix} \begin{bmatrix} (l_{13}^d - l_{13})^{\frac{p}{q}} \\ (l_{23}^d - l_{23})^{\frac{p}{q}} \end{bmatrix}, \tag{4.16}$$

which is identical to a terminal attractor model where (p,q) can be, for example, set to (2,3).

Therefore

$$\begin{bmatrix} \dot{l}_{13} \\ \dot{l}_{23} \end{bmatrix} = \begin{bmatrix} \alpha_1^* & 0 \\ 0 & \alpha_2^* \end{bmatrix} \begin{bmatrix} (l_{13}^d - l_{13})^{\frac{p}{q}} \\ (l_{23}^d - l_{23})^{\frac{p}{q}} \end{bmatrix} = \alpha^* . l_e , \tag{4.17}$$

where

$$\alpha^* = \begin{bmatrix} \alpha_1^* & 0 \\ 0 & \alpha_2^* \end{bmatrix}, \quad l_e = \begin{bmatrix} (l_{13}^d - l_{13})^{\frac{2}{3}} \\ (l_{23}^d - l_{23})^{\frac{2}{3}} \end{bmatrix}.$$

By arbitrary setting reaching time T_r, we obtain a formula for finding α_1^*, α_2^* as follows.

$$\begin{aligned} \alpha_1^* &= \left(l_{13}^d - l_{13}(0)\right)^{\frac{1}{3}} / \left(\frac{T_r}{3}\right) = sign(l_{23}^d - l_{23}(0)) \left| l_{23}^d - l_{23}(0) \right|^{\frac{1}{3}} / \left(\frac{T_r}{3}\right), \\ \alpha_2^* &= \left(l_{23}^d - l_{23}(0)\right)^{\frac{1}{3}} / \left(\frac{T_r}{3}\right) = sign(l_{23}^d - l_{23}(0)) \left| l_{23}^d - l_{23}(0) \right|^{\frac{1}{3}} / \left(\frac{T_r}{3}\right), \end{aligned} \tag{4.18}$$

where functions $sign(.)$ and $|.|$ are used to avoid complex solutions.

From (4.8) and (4.17) we can generate a feedback control law as follows.

$$u_3 = B_u^{-1}(v_u + \alpha^* . l_e), \quad 0 \le t \le T_r, \tag{4.19}$$

where $$B_u^{-1} = \frac{1}{D\sin(\gamma_2 - \gamma_1)} \begin{bmatrix} D\sin\gamma_2 & -D\sin\gamma_1 \\ -\cos\gamma_2 & \cos\gamma_1 \end{bmatrix}.$$

The solution of the system is

$$l_{i3} = l_{i3}^d - \left(\left(l_{i3}^d - l_{i3}(0)\right)^{\frac{1}{3}} - \frac{\alpha_i^*}{3} t \right)^3, \quad 0 \le t \le T_r, i = 1,2, \tag{4.20}$$

which monotonically converges to desired value l_{i3}^d within a finite time T_r.

Zero dynamics in term of θ_3 can also be proven to be stable [Jongusuk and Mita, 2001].

Remarks

1. We can find out other alternatives to choose p and q. Let us firstly present another proposition on a terminal attractor model similar to the one above.

Proposition 4.2 *The origin of a differential equation*

$$\dot{z} = -\lambda z^{p/q}, \quad \lambda > 0 \tag{4.21}$$

is a terminal attractor, with a finite reaching time

$$T_r = \frac{z(0)^{1-\frac{p}{q}}}{\lambda(1-\frac{p}{q})}, \tag{4.22}$$

where $(p,q) \in \Omega$ *with*

$$\Omega = \left\{ (p,q) \middle| p = 2m-1, q = 2n-1, m < n, m,n \in N^+ \right\} \text{ and } N^+ = N \setminus \{0\}$$

The proof of this proposition is similar to the one of Proposition 4.1.

Now we can set

$$\begin{bmatrix} \dot{l}_{13} \\ \dot{l}_{23} \end{bmatrix} = \begin{bmatrix} \alpha_1^* & 0 \\ 0 & \alpha_2^* \end{bmatrix} \begin{bmatrix} (l_{13}^d - l_{13})^{\frac{p}{q}} \\ (l_{23}^d - l_{23})^{\frac{p}{q}} \end{bmatrix} = \alpha^* . l_e ,$$

where (p,q) can be , for example, set to (1,3), hence

$$\alpha^* = \begin{bmatrix} \alpha_1^* & 0 \\ 0 & \alpha_2^* \end{bmatrix}, \quad l_e = \begin{bmatrix} (l_{13}^d - l_{13})^{\frac{1}{3}} \\ (l_{23}^d - l_{23})^{\frac{1}{3}} \end{bmatrix}.$$

In this case, by arbitrary setting reaching time T_r, we obtain

$$\alpha_1^* = \left(l_{13}^d - l_{13}(0)\right)^{\frac{2}{3}} / \left(\frac{2T_r}{3}\right) = \left|l_{13}^d - l_{13}(0)\right|^{\frac{2}{3}} / \left(\frac{2T_r}{3}\right),$$

$$\alpha_2^* = \left(l_{23}^d - l_{23}(0)\right)^{\frac{2}{3}} / \left(\frac{2T_r}{3}\right) = \left|l_{23}^d - l_{23}(0)\right|^{\frac{2}{3}} / \left(\frac{2T_r}{3}\right).$$

The control law (4.19) give the solution of the system

$$\left(l_{i3}^{d}-l_{i3}\right)^{2}=\left(\left(l_{i3}^{d}-l_{i3}(0)\right)^{\frac{2}{3}}-\frac{2\alpha_{i}^{*}}{3}t\right)^{3},\quad 0\le t\le T_{r},\ i=1,2, \tag{4.23}$$

which also ensure that l_{i3} monotonically converges to desired value l_{i3}^{d} within a finite time T_r.

2. As in the original *l-l* control, a singular case is found when $\gamma_2-\gamma_1=k\pi,\ (k\in Z)$. This is the case when matrix B_u is singular, and the three robots lie on the same line connecting their reference points.

The above-mentioned control laws all have a singularity when the three robots lie on the same line connecting them (see Figure 4.3). This singularity makes existing controllers unable to completely solve the initialisation problem in formation control for a group of mobile robots. A new approach will be proposed in the next section to deal with the singularity.

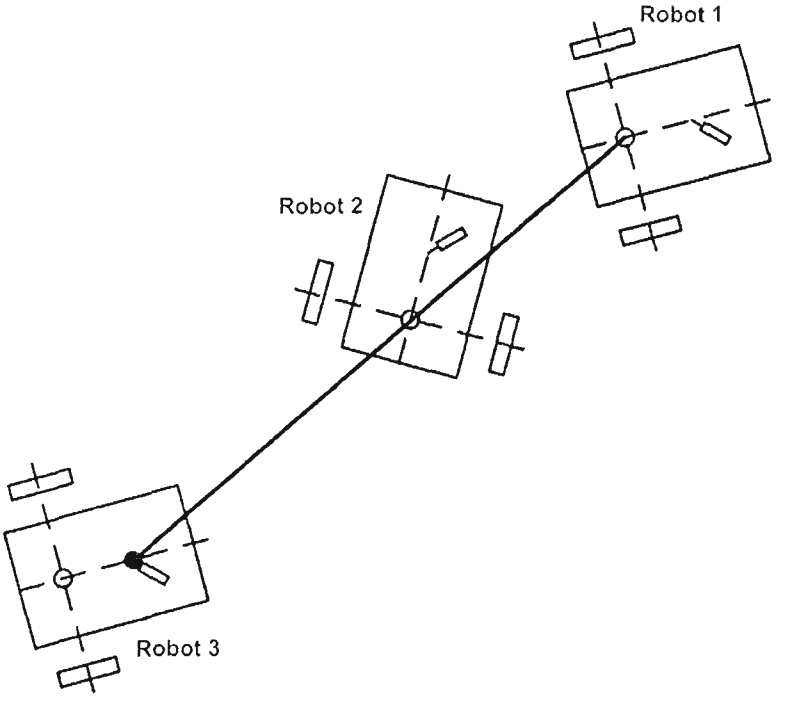


Figure 4.3 Singularity in *l-l* control

4.4 Three-point *l-l* control

To deal with the singularity problem, three *l-l* controllers are proposed here, taking into account the distances to robot 1 and robot 2 from three virtual points. These points are located around the centre of robot 3 to form a certain triangle. An appropriate *l-l* controller shall be chosen in the case of singularity.

Let us firstly introduce a proposition about velocities of a virtual robot [Nguyen *et al.*, 2006b].

Proposition 4.3 (*Virtual Robot velocities) A virtual robot of robot* i, *having predefined values of* $R = R^*, L = 0$, *can be considered apparently as an "independent" robot with velocities* $v_i' = v_i + R^*\omega_i$ *and* $\omega_i' = \omega_i$, *where* v_i *and* ω_i *are velocities of robot i.*

Proof

Indeed, by considering a virtual robot of robot i with pre-defined clearances $R = R^*, L = 0$, one has from (3.30)

$$\begin{cases} x_{vi} = x_i + R^* \sin\theta_i \\ y_{vi} = y_i - R^* \cos\theta_i \\ \theta_{vi} = \theta_i, \end{cases} \tag{4.24}$$

and hence,

$$\begin{cases} \dot{x}_{vi} = \left(v_i + R^*\omega_i\right)\cos\theta_i \\ \dot{y}_{vi} = \left(v_i + R^*\omega_i\right)\sin\theta_i \\ \dot{\theta}_{vi} = \dot{\theta}_i = \omega_i. \end{cases} \tag{4.25}$$

The virtual velocities of robot i given in Proposition 4.3 can then be obtained by comparing (4.25) to model (4.1).

■

Consider now the case when distances to robot 1 and robot2 are from point K, which is different from the centre of robot 3 and determined by distances r_K, l_K, as shown in Figure 4.4. Here, K is considered as a head point located, along the longitudinal axis, at distance l_K from the centre of a virtual robot R_{v3} of robot 3, defined with $R = r_K$, $L = 0$. According to Proposition 4.3, the velocities of virtual robot R_{v3}:

$$\begin{cases} v_{v3} = v_3 + r_K \omega_3 \\ \omega_{v3} = \omega_3. \end{cases} \tag{4.26}$$

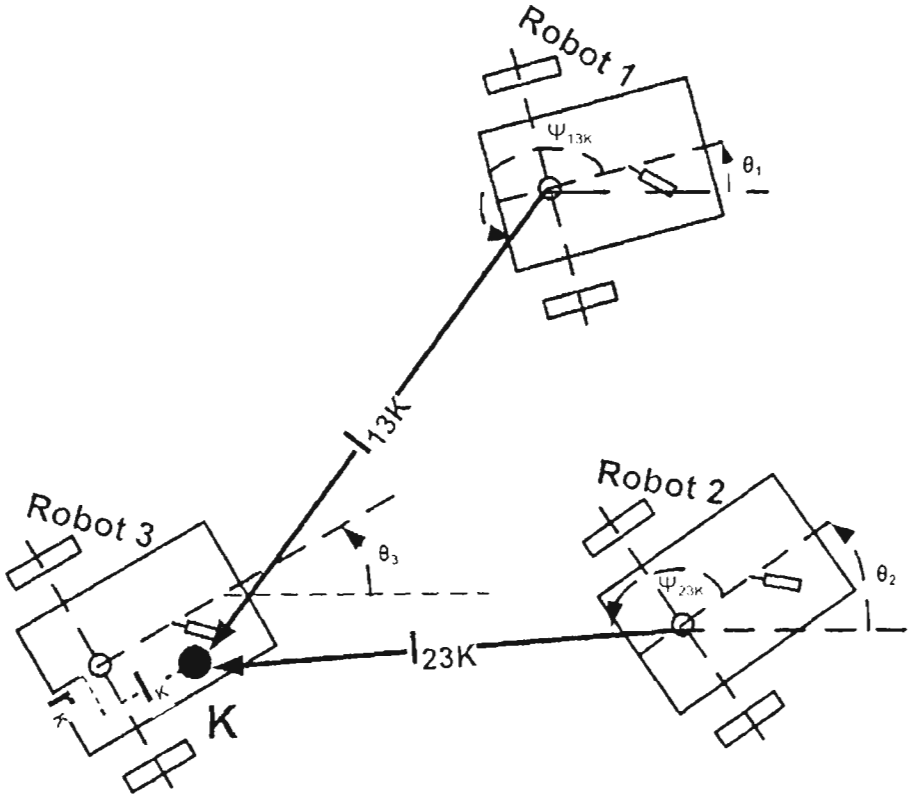


Figure 4.4: *l-l* control with respect to virtual point K

The kinematic model for R_{v3} under *l-l* control is

$$\begin{bmatrix} \dot{l}_{13K} \\ \dot{l}_{23K} \end{bmatrix} = \begin{bmatrix} \cos\gamma_{1K} & l_K \sin\gamma_{1K} \\ \cos\gamma_{2K} & l_K \sin\gamma_{2K} \end{bmatrix} \begin{bmatrix} v_{v3} \\ \omega_{v3} \end{bmatrix} - \begin{bmatrix} v_1 \cos\psi_{13K} \\ v_2 \cos\psi_{23K} \end{bmatrix} = B_{uK} u_{v3} - v_{uK}, \tag{4.27}$$

and

$$\dot{\theta}_{v3} = \omega_{v3}, \tag{4.28}$$

where $B_{uK} = \begin{bmatrix} \cos\gamma_{1K} & l_K \sin\gamma_{1K} \\ \cos\gamma_{2K} & l_K \sin\gamma_{2K} \end{bmatrix}$, $u_{v3} = \begin{bmatrix} v_{v3} \\ \omega_{v3} \end{bmatrix}$, $v_{uK} = \begin{bmatrix} v_1 \cos\psi_{13K} \\ v_2 \cos\psi_{23K} \end{bmatrix}$.

Therefore, similarly to the law proposed in [Jongusuk and Mita, 2001], one can apply the following control law for virtual robot R_{v3}:

$$u_{v3} = \begin{bmatrix} v_{v3} \\ \omega_{v3} \end{bmatrix} = B_{uK}^{-1}(v_{uK} + \alpha_K^* . l_{eK}), 0 \le t \le T_r ; \tag{4.29}$$

where

$$\begin{cases} \gamma_{iK} = \theta_i + \psi_{i3K} - \theta_3, (i = 1,2) \\ l_{eK} = \begin{bmatrix} \left(l_{13K}^d - l_{13K}\right)^{\frac{2}{3}} \\ \left(l_{23K}^d - l_{23K}\right)^{\frac{2}{3}} \end{bmatrix}, \alpha_K^* = \begin{bmatrix} \alpha_{1K}^* & 0 \\ 0 & \alpha_{2K}^* \end{bmatrix} \\ \alpha_{1K}^* = sign(l_{13K}^d - l_{13K}(0)) \left| l_{13K}^d - l_{13K}(0) \right|^{\frac{1}{3}} / \left(\frac{T_r}{3}\right) \\ \alpha_{2K}^* = sign(l_{23K}^d - l_{23K}(0)) \left| l_{23K}^d - l_{23K}(0) \right|^{\frac{1}{3}} / \left(\frac{T_r}{3}\right), \end{cases} \tag{4.30}$$

and $B_{uK}^{-1} = \dfrac{1}{l_K \sin(\gamma_{2K} - \gamma_{1K})} \begin{bmatrix} l_K \sin\gamma_{2K} & -l_K \sin\gamma_{1K} \\ -\cos\gamma_{2K} & \cos\gamma_{1K} \end{bmatrix}$.

From (4.26), the control law in terms of velocities for robot 3 can be calculated as

$$u_3 = \begin{bmatrix} v_3 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} v_{v3} - r_K \omega_{v3} \\ \omega_{v3} \end{bmatrix}. \tag{4.31}$$

As stated, the three *l-l* controllers are to be switched correspondingly, for example, to three virtual head points of robot 3, namely A, B, and C, selected to form the shape of a triangle, as shown in Figure 4.5. By that way, the singularity problem associated with *l-l* control, i.e. when the head point of robot 3 is aligned with the line connecting two other robots, can be completely overcome. Triggering the switching in practice depends on the level of sensitivity of the robot actuator control voltage with respect to singularities.

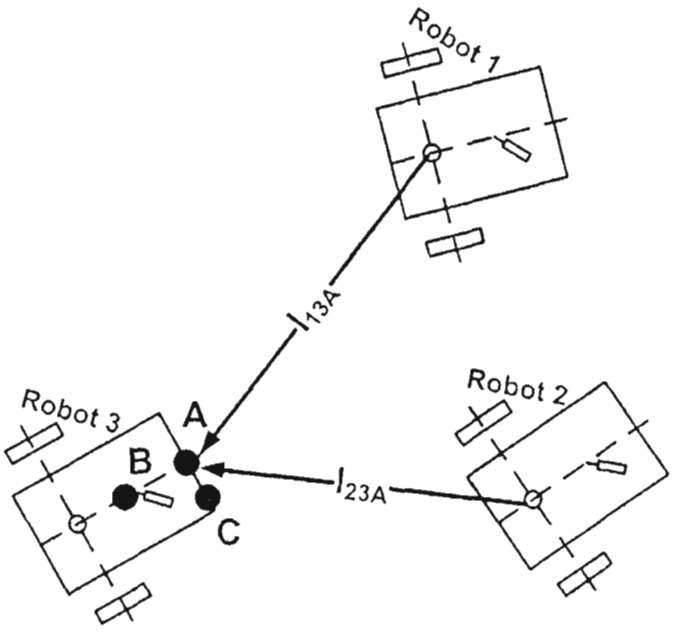


Figure 4.5: Switching among three *l-l* controllers

Note that for the proposed 3PLL control, there shall be a parameter constraint which can be described as:

$$(l_{i3}^d)^2 - (r_K^2 + l_K^2) > (2r_{safe})^2, \quad i = 1,2; \; K \in \{A, B, C\}, \tag{4.32}$$

where $2r_{safe}$ is the safe distance between two centre points of two robots.

Condition (4.32) allows for a safe distance between the robots under 3PLL control. Furthermore, the reference distances l_{13}^d and l_{23}^d in *l-l* control should be designed in order to satisfy that the group motion is accomplished in a restricted area that is limited by the communication range among robots

4.5 Reactive control scheme

In the tracking phase, if the safe distance given (3.41) is not preserved, VHRT control cannot ensure the collision avoidance between two concerning robots. A reactive control scheme is then proposed for this purpose, based on the general notice that 3PLL control should be used for the robot of lower priority in the case of potential collision detected, to drive the follower to diverge but head to target position so that collision will

most likely not happen after switching back to tracking control. In this case the follower, robot 3, must refer to two leaders: robot 1, which is the robot of higher priority as the first leader, and robot 2, which is the virtual robot of robot 1 as the second leader. The virtual robot's predefined clearances with respect to its host are then $R = 2r_{safe} + D_{\max}$ or $R = -(2r_{safe} + D_{\max})$ and $L = 0$, where $D_{\max}$ is the largest distance from one of three head points of robot 3 to its centre. The sign of clearance R is decided such that the second leader should be close to the target position of robot 3. To illustrate, Figure 4.6 depicts the case $R = -(2r_{safe} + D_{\max})$ and $L = 0$, where robot 3 of the lowest priority is to switch to 3PLL control, robot 1 is the first leader, and its virtual robot, robot 2. The control parameters l_{13}^d and l_{23}^d are designed as follows,

$$\begin{aligned} l_{13}^d &= 2r_{safe} + D_{\max} + \delta_1, \\ l_{23}^d &= \delta_2. \end{aligned} \tag{4.33}$$

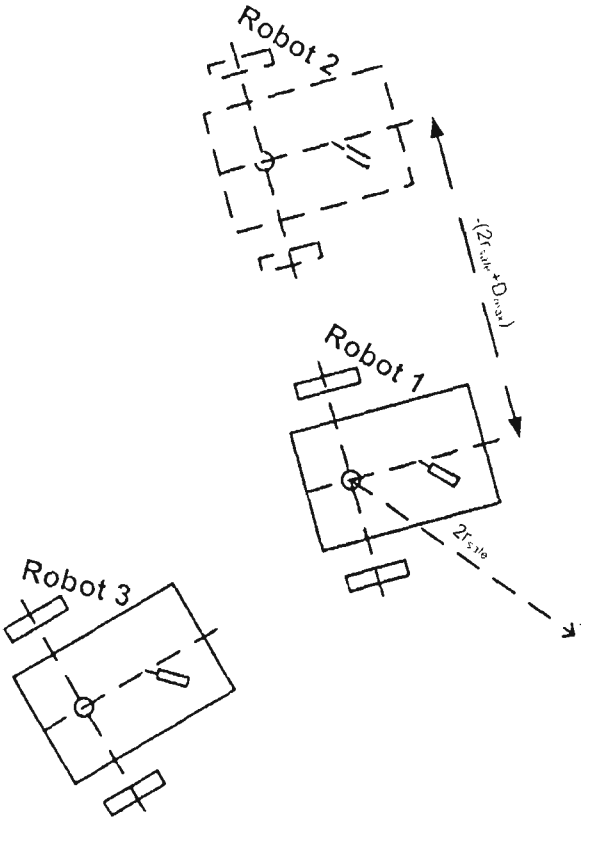


Figure 4.6: Case $R = -(2r_{safe} + D_{\max})$ with 3PLL control

When using 3PLL control, the distances from one of three head points of robot 3 to the centres of its leaders are taken into account, l_{13}^d has to be therefore increased by $D_{\max}$. Margins δ_1 and δ_2 are deliberately augmented to l_{13}^d and l_{23}^d to ensure the distance between centres of robot 1 and robot 3 to be strictly greater than $2r_{safe}$. These together will help to drive robot 3 closer to the virtual robot 2 while going around a safe boundary of robot 1, which is a circle with centre of robot 1 and radius $2r_{safe}$.

Note that this reactive control scheme is similar to the ones proposed in [Ha *et al.*, 2005] but the velocities of the virtual robot can be easily determined based on those of robot 1 according to Proposition 4.3. They are required for the *l-l* control for robot 3

4.6 Proposed algorithm for combination of VHRT and 3PLL control

A simple algorithm is used here to combine the VHRT and 3PLL control (see Figure 4.7).

Let us consider robot i ($i = 2,3$) in the group. First, parameters of controllers must be initialised for formation establishment, taking into account the desired position of this robot in relation to its leader, and for collision avoidance control, considering safe distances to its two nearest robots. Then possibility of collision is checked between two followers (robot 2 and robot 3). If potential collision is detected, the robot compares its priority level to that of the other follower. Upon a lower priority level (higher index), i.e. $i = 3$, it has to change to the 3PLL control for collision avoidance. If no possibility of collision exists or the priority level is higher than that of the relevant robot (i.e. $i = 2$), robot i will use VHRT control to form and maintain the required formation.

Note that in the tracking phase if there is a possibility of collision between the robot and its leader (i.e. the leader is found in "critical area" at the beginning of this phase), proposed reactive control scheme will be used firstly for collision avoiding. After

avoiding collision by using the reactive control scheme, the robot will switch back to VHRT control to eventually establish the desired configuration.

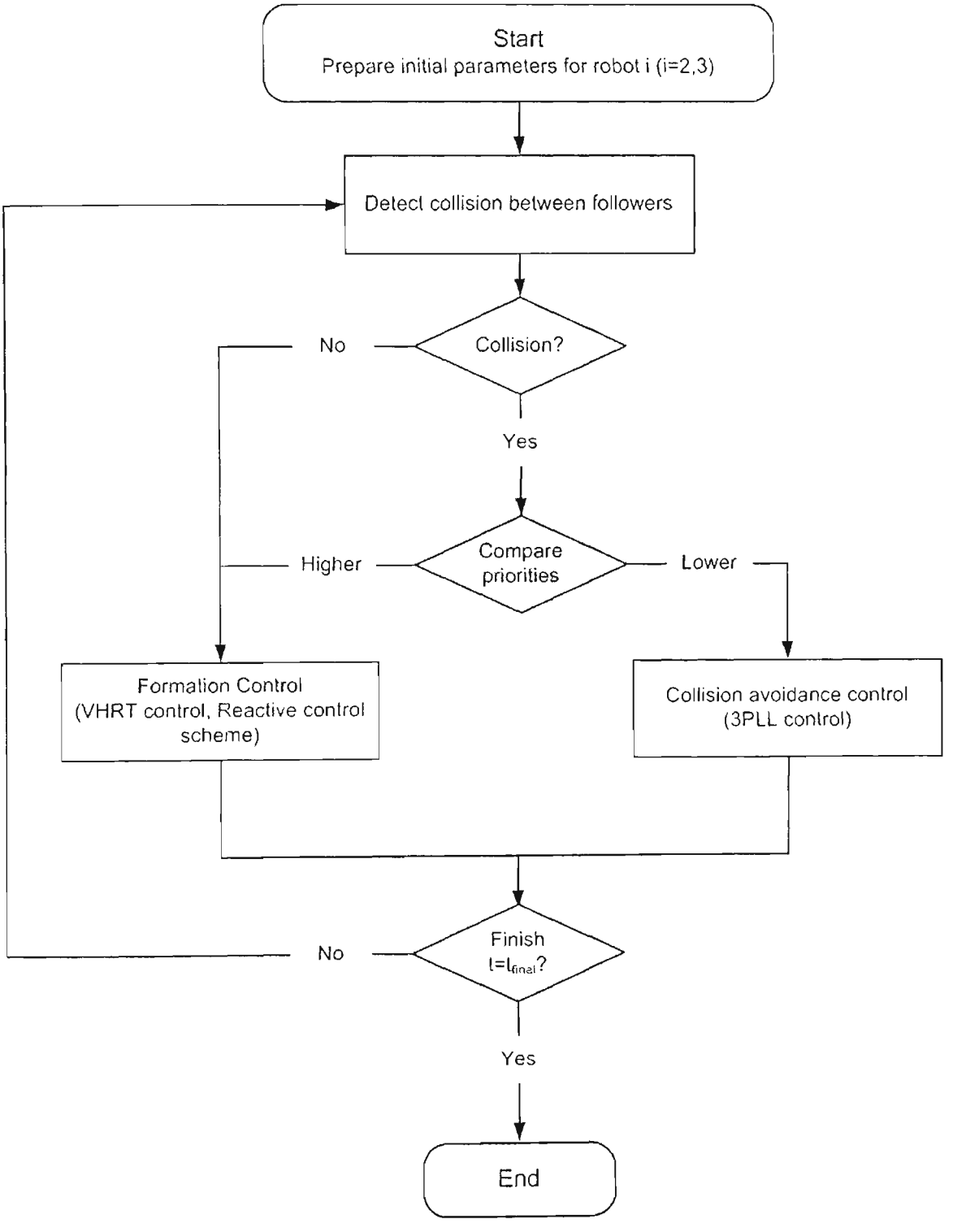


Figure 4.7: Algorithm to combine VHRT and 3PLL control

The check of possibility of collision should be repeatedly done in a sufficiently small sampling time to guarantee that any potential collision can be detected and avoided in time.

Path planning for the leader of the group (i.e. robot 1) should be performed considering the predefined formation configuration and the given workspace but this problem is beyond the scope of this thesis.

4.7 Simulation results

We present here two simulations. The first simulation illustrates the capability of forming a line formation, whilst the second one shows the way 3PLL control is applied for the case of singularities.

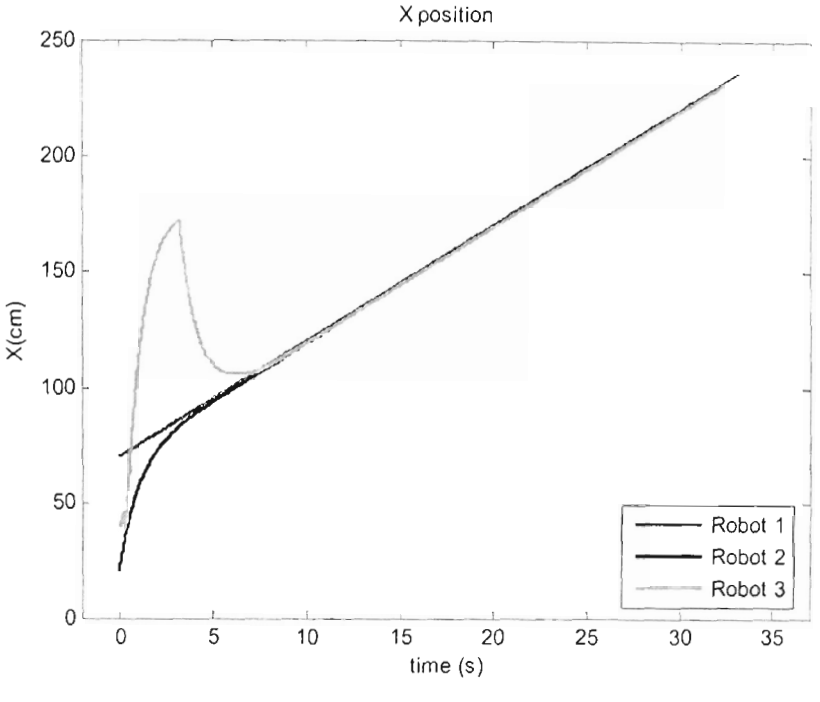
The number of occlusion in fact can be decreased by modifying l_{13}^d, l_{23}^d and T_r in each occurrence, here we simply use constant values for all circumstances.

4.7.1 Simulation 4.1: Line formation

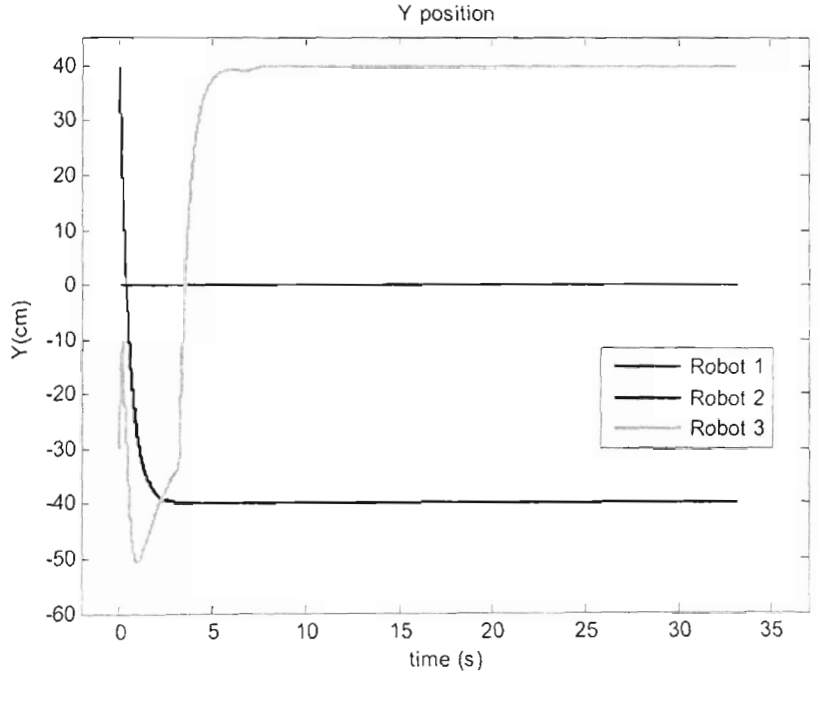
To illustrate the capability of forming a line formation and avoiding collision among robots in a formation, let us first consider the case of three mobile robots initialized from arbitrary positions. Parameters and conditions used in this simulation were set as in Table 4.1.

Figures 4.8 and 4.9 show the time responses of position X, Y, orientation θ and the trajectories of two mobile robots in the global coordinates. With these initial conditions, potential collision could be observed at time point $t = 0.2s$ between robot 3 and robot 2. At that time, robot 3 applied *l-l* control with head point A to avoid collisions and then switched back to VRTH control until the desired line formation is obtained.

The simulation results illustrate that the three robots can successfully get into and maintain a line formation without collision among them.



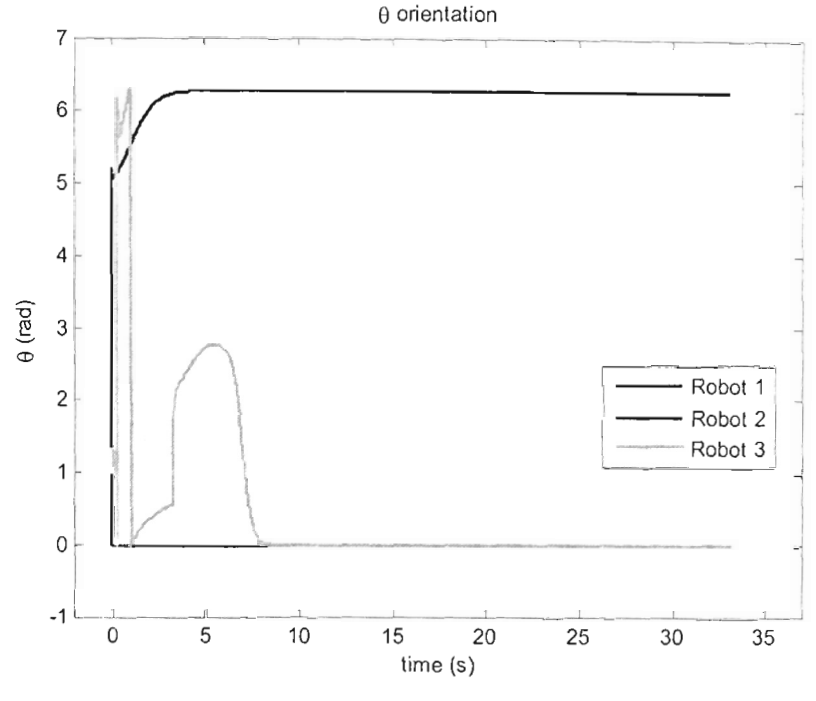
a)



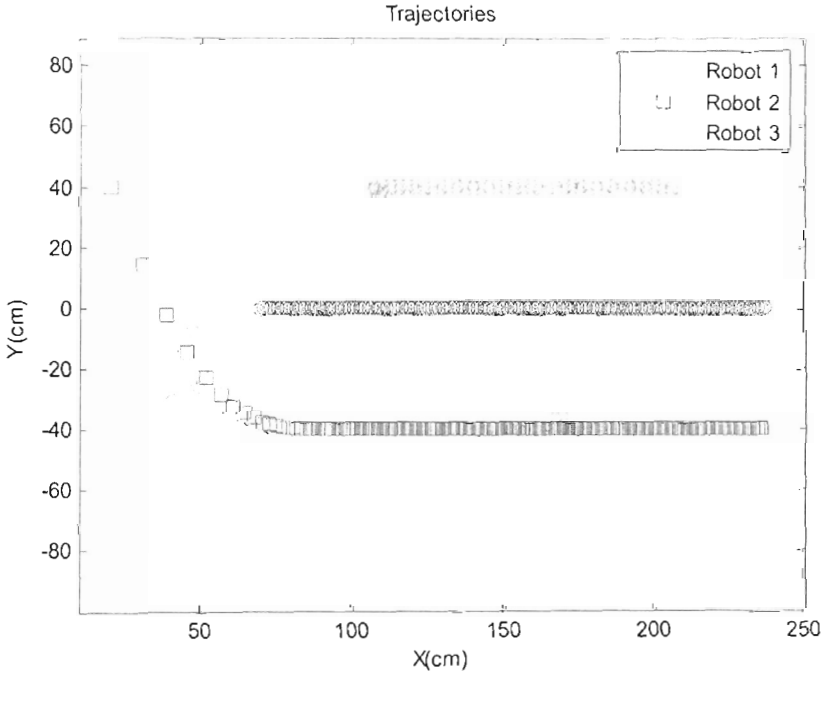
b)

Figure 4.8: Simulation 4.1 results - X, Y position

a) X position b) Y position



a)



b)

Figure 4.9: Simulation 4.1 results – θ orientation and trajectories

a) θ- orientation b) Trajectories

Table 4.1: Parameters and Conditions for Simulation 4.1

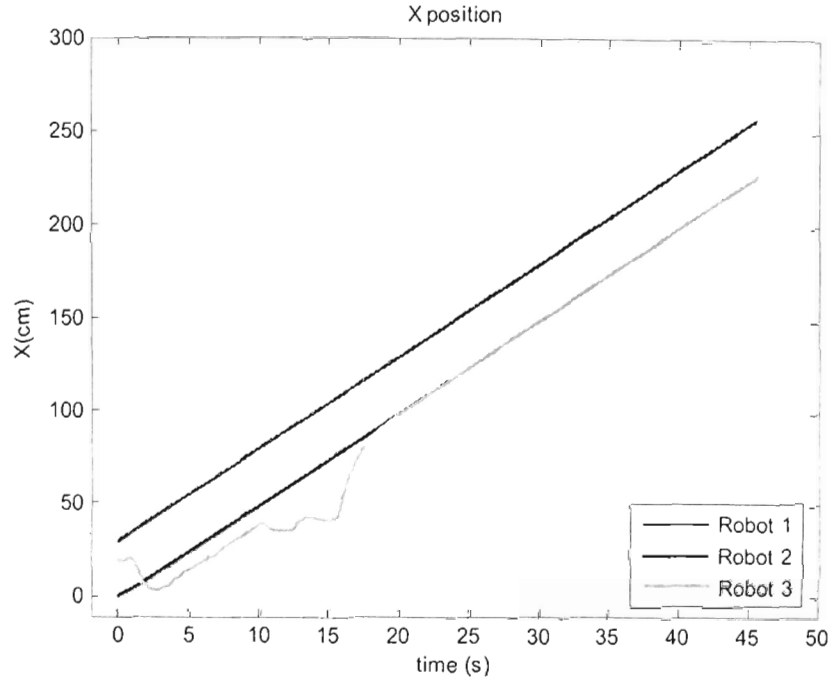
Simulation Parameters		Robot 1	Robot 2	Robot 3
Initial Conditions	$x(0)$ (cm)	70	20	40
	$y(0)$ (cm)	0	40	-30
	$\theta(0)$ (rad)	0	0	1
	v (cm)	5		
	ω (rad/s)	0		
Parameters for desired formation	R (cm)		40	-40
	L (cm)		0	0
Tracking margin for head robot	d (cm)		1	1
Safe distance between any two robots	$2r_{safe}$ (cm)	26	26	26
Parameters for VHRT control	λ_1 (s^{-1})		1	1
	λ_2 (s^{-1})		2	2
Parameters for *l-l* control with point A	r_A (cm)			0
	l_A (cm)			12
	T_r (s)			3
	l^d_{13A} (cm)			100
	l^d_{23A} (cm)			100

4.7.2 Simulation 4.2: Case of singularities

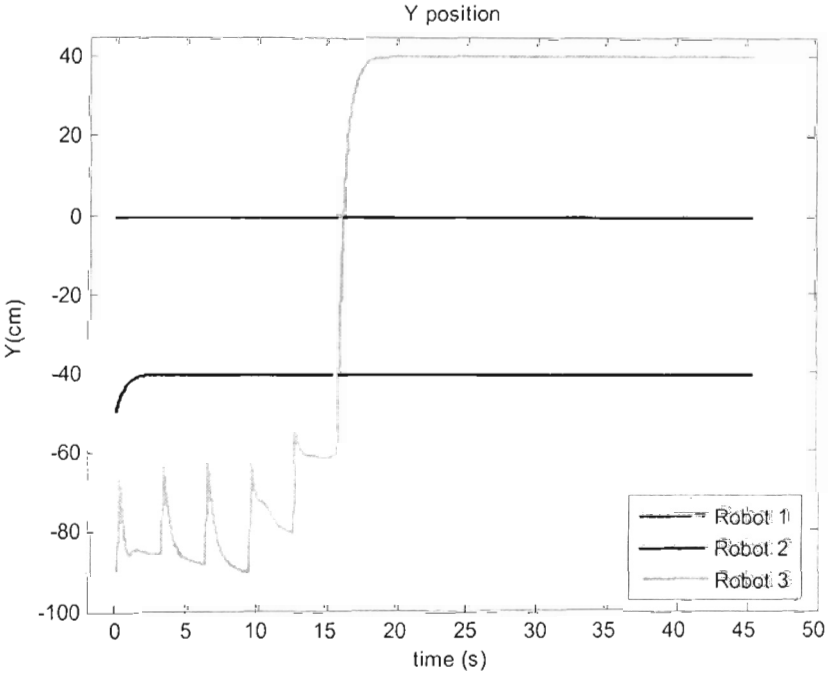
To illustrate the capability of avoiding singularities and possibilities of collision among robots in a formation, we choose the case of three mobile robots moving to form a wedge formation to simulate. Parameters and conditions used in this simulation were set as in Table 4.2.

Figures 4.10 and 4.11 show the time responses of position X, Y, orientation θ and the trajectories of two mobile robots in the global coordinates. With these initial conditions, potential collision could be observed at time point $t = 0.2s; 3.21s; 6.33s; 9.46s$ and 12.6s when using *l-l* controller with head point A of robot 3. It is observed that at $t = 9.46s$, due to singularity, the system switched to the *l-l* controller with respect to head point B.

Our simulation results illustrate that the three robots can successfully get into and maintain a wedge formation without inter-robot collision even in the presence of singularities.



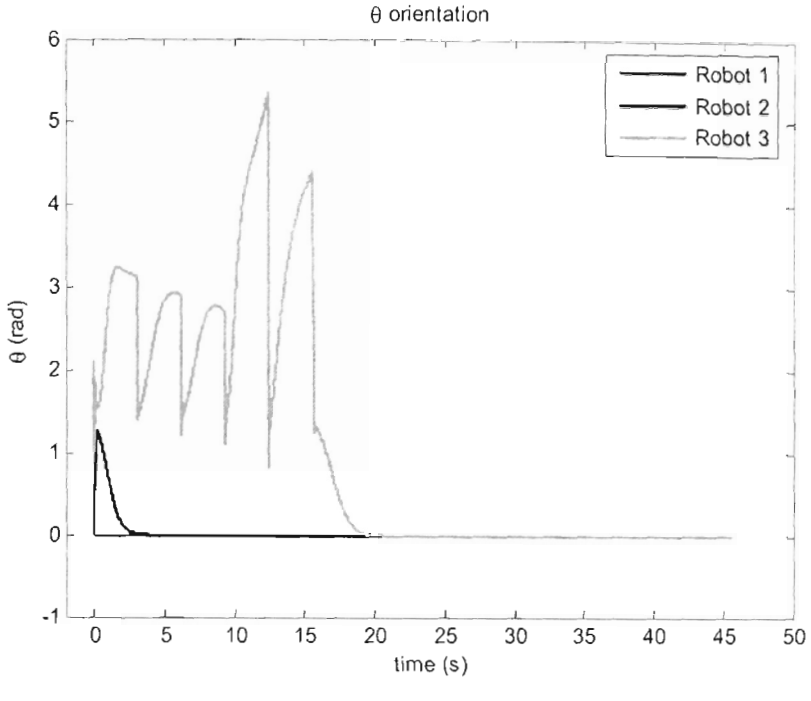
a)



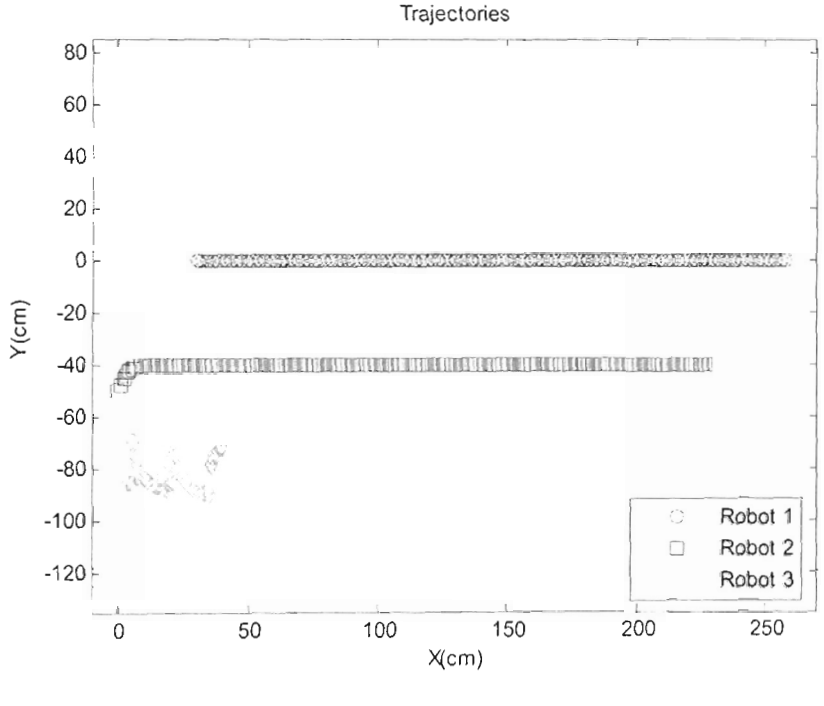
b)

Figure 4.10: Simulation 4.2 results - X, Y position

a) X position b) Y position



a)



b)

Figure 4.11: Simulation 4.2 results – θ orientation and trajectories

a) θ- orientation b) Trajectories

Table 4.2: Parameters and Conditions for Simulation 4.2

Simulation Parameters		Robot 1	Robot 2	Robot 3
Initial Conditions	$x(0)$ (cm)	30	0	20
	$y(0)$ (cm)	0	-50	-90
	$\theta(0)$ (rad)	0	0	1
	v (cm)	5		
	ω (rad/s)	0		
Parameters for desired formation	R (cm)		40	-40
	L (cm)		30	30
Tracking margin for head robot	d (cm)		1	1
Safe distance between any two robots	$2r_{safe}$ (cm)	26	26	26
Parameters for VHRT control	λ_1 (s^{-1})		1	1
	λ_2 (s^{-1})		2	2
Parameters for *l-l* control with point A	r_A (cm)			0
	l_A (cm)			12
	T_r (s)			3
	l^d_{13A} (cm)			100
	l^d_{23A} (cm)			50
Parameters for *l-l* control with point B	r_B (cm)			0
	l_B (cm)			6
	T_r(s)			3
	l^d_{13B} (cm)			100
	l^d_{23B} (cm)			50

4.8 Conclusion

This chapter has presented a new *three point l-l* control to avoid inter-robot collisions in a group of three mobile robots with the capability of singularity alleviation. Combining this control law with the VHRT one, which is presented in the previous chapter, a new algorithm has been proposed incorporated with a reactive control scheme for formation control of a group of mobile robots.

There is a trade-off in choosing of three virtual points A, B, C around the centre point of robot 3. They should be separated from each other far enough to ensure that the singularity can be overcome when switching among three concerning controllers. On the other hand, they should be close enough to each other so that the switching does not

cause any sudden change of velocities of robot 3. In addition, from equation (4.29) it is clear that one should choose l_{13}^d, l_{23}^d and T_r appropriately, that is l_{13}^d, l_{23}^d to be small enough and T_r to be large enough, to alleviate the sudden change of velocities of robot 3 when switching from VHRT to 3PLL control.

The proposed approach in this chapter could be an appropriate solution to the initialisation problem for a group of three mobile robots. For a group of more than three robots, a formation can be established firstly for pre-defined three robots and then enlarged gradually for other robots in the group. Such a step-by-step algorithm is expected to reduce the possibilities of collisions among robots in the group. A procedure for formation initialisation of a group of N mobile robots is the content of the next chapter.

Chapter 5

Formation initialisation for a group of N mobile robots

5.1 Introduction

The formation initialisation problem involves the process of deploying mobile robots in the group to assemble from arbitrary initial points to establish a desired formation. Collision avoidance among robots and between robots and obstacles is always a great challenge in robotic formation control, especially in the initialisation phase when the group of mobile robots has not formed the desired geometrical shape.

In the robotics literature, there has been little effort paid to the formation initialization issue in the research of multi-robot coordination. In [Sugihara and Suzuki, 1996], a group of simulated robots formed approximations to circles and simple polygons, using global knowledge of all robots' positions. Each robot oriented itself to, e.g., the furthest and nearest robot. In [Chen and Luh, 1994], a similar setup was presented, but group motion was also considered, e.g., a matrix formation performing a right turn. A formation is defined by a *Virtual Structure* (VS) in [Lewis and Tan, 1997]. The algorithm was iteratively used to command the VS to the current positions, displace the VS in some desired direction, and update the robots' positions.

All above approaches require that all robots had global knowledge about the group. In [Fredslund and Mataric, 2002] a general algorithm was proposed for N mobile robots to establish and maintain some predetermined geometric shape using only local sensing and minimal communication. The capability of forming stable, robust and switchable formations however was usually demonstrated by extensive simulations and experiments.

The simple algorithm proposed in the previous chapter, which combines VHRT and 3PLL control incorporated with a reactive control scheme, can be an appropriate approach for initialization formation for a group of three mobile robots. For a group of more than three mobile robots, collisions among robots may easily happen in the initialization phase. In this chapter a step-by-step procedure incorporated VHRT, 3PLL control with reactive control schemes will be proposed to ensure that a desired formation of a group of N mobile robots can be established and maintained without inter-robot collisions. We will also use graph concept to model formations.

Section 5.2 presents models and problem formulation. Some reactive control schemes to avoid collisions among robots are presented in Section 5.3. A step-by step procedure to initialise and maintain a desired formation for a group of N mobile robots is given in Section 5.4. Section 5.5 describes control graphs for modelling formations and enumerates all allowable graphs for a group of N mobile robots. Simulation results to validate the proposed approach are presented in Section 5.6. A conclusion is given in Section 5.7.

5.2 Model and problem formulation

The kinematic model (3.1) or (4.1) as in previous chapters is used here for a three-wheel mobile robot.

The possibility of collision between any two mobile robots i and j is also measured by using function f_{ij} as in (3.2) and (3.3).

In addition to four assumptions presented in the previous chapter, a new one is required to reduce the possibility of inter-robot collision in a group of N mobile robots.

- ***Assumption 5***: Distance between any two robots in desired formations should preserve a certain lower limit, which is double of the safe distance between them, i.e. $4r_{safe}$.

The objective here is to coordinate N mobile robots to achieve:

1. any desired formation which satisfies Assumption 5, and
2. no collision between any two mobile robots.

5.3 Reactive control schemes

A procedure is proposed here using VHRT control as a basic tracking controller for forming a desired geometrical shape of a group of multiple mobile robots. Collisions among robots in the group can be avoided by using 3PLL control with appropriate reactive control schemes. The idea is that should collision occur among robots according to the collision detection criteria described by function f_{ij}, 3PLL control will be used to drive the lowest priority robot (i.e. robot with the highest index) among concerning robots to diverge from them but still heading to target position so that collision will most unlikely happen after switching back to VHRT control.

Generally, cases necessitating reactive control can be dealt with by using two following schemes. The first scheme is the same to the one described in previous chapter and is restated here for the sake of completeness.

5.3.1 Scheme 1: Potential collision between two robots

This scheme is used for a potential collision between any two robots, as illustrated in Figure. 5.1. The lower priority robot, robot 3, will switch to 3PLL control with respect to two leaders: the higher priority robot as the first leader (robot 1) and a VR of this robot as the second leader (robot 2) with predefined clearances $R = 2r_{safe} + D_{\max}$ (robot

2') or $R = -(2r_{safe} + D_{\max})$ (robot 2") and $L = 0$. Here, $D_{\max}$ is the largest distance from one of three head points of robot 3 to its centre, and the sign of clearance R (i.e. the choice robot 2 as robot 2' or robot 2") is decided such that the second leader should be close to the target position of robot 3.

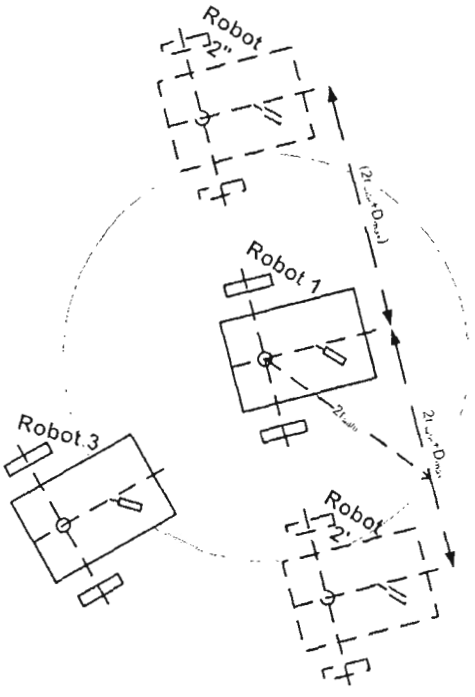


Figure 5.1: Reactive control scheme 1 with 3PLL control

The control parameters l_{13}^d and l_{23}^d can be designed as follows

$$\begin{aligned} l_{13}^d &= 2r_{safe} + D_{\max} + \delta_1, \\ l_{23}^d &= \delta_2. \end{aligned} \tag{5.1}$$

Using 3PLL control with the above parameters will drive robot 3 closer to the VR (robot 2) while going around the safe boundary of robot 1, which is a circle with centre of robot 1 and radius $2r$. This will obviously reduce the possibility of robot 3 to collide with the other robots. The reason that l_{13}^d is $2r_{safe} + D_{\max}$ rather than $2r_{safe}$ is that in 3PLL control, the distances from one of three head points of robot 3 to the centres of its leaders is referred instead of the distances among their centres. Thus in order to ensure collision avoidance, l_{13}^d has to be increased by the largest distance from one of three

head points of robot 3 to its centre. Margins δ_1 and δ_2 are deliberately augmented to l_{13}^d and l_{23}^d to ensure the distance between centres of robot 1 and robot 3 to be strictly greater than $2r_{safe}$.

5.3.2 Scheme 2: Potential collision between three robots

This scheme is used when there is a possibility of collision between a robot and two other robots. The lowest priority robot will then apply 3PLL control with respect to two leaders, which are the two other robots concerned. The control parameters l_{13}^d and l_{23}^d are proposed as,

$$\begin{aligned} l_{13}^d &= 2r_{safe} + D_{\max} + \delta_1, \\ l_{23}^d &= 2r_{safe} + D_{\max} + \delta_2, \end{aligned} \tag{5.2}$$

where again, $D_{\max}$, δ_1 and δ_2 are augmented to l_{13}^d and l_{23}^d for the same reason as explained above.

5.4 Formation initialisation procedure

A step-by-step procedure is proposed in this section for initialising a group of N mobile robots to enter a desired formation [Nguyen *et al.*, 2006c]. At each step, robots in the group are classified as *active* or *inactive*. *Active* robots will participate to establish the formation while *inactive* robots stay at its initial position. The process of formation initialisation will then run until all robots in the group become *active* and a desired formation is obtained. In addition, in the proposed procedure, each robot can play the role of a follower (tracking another robot) or of a leader (guiding another one). The leader of the whole group is called the *lead* robot and is indexed by 1 [Balch and Arkin, 1995].

We also assume that *inactive* robots are designated not to obstruct any *active* robot. This condition can be satisfied by appropriately numbering and choosing robots to be *active* at steps.

The proposed process of formation initialisation can now be performed with the following algorithm (see Figure 5.2).

1. All robots in the group are initially considered *inactive*. Choose the *lead* robot, indexed by i=1, to guide the whole group. Let it become *active*.
2. Index (or reindex) all *inactive* robots from (i+1) to N, based on their initial position with respect to the motion of the *lead* robot.
3. Let one or two *inactive* robots with smallest indices become *active*. Use reactively-controlled VHRT-3PLL to get them into desired positions while avoiding collision with other robots until all i *active* robots have reached their positions in the group. Go to 2.

   If there is no *inactive* robot left (or i=N), the desired formation has established.
4. Exit.

The formation initialisation phase is a multi-step process. Each step begins when one or two *inactive* robots become *active* and finishes when all current *active* robots reach their desired positions in the group. Steps are indexed by k with $1 \le k \le N$. The number of steps n_k of the initialisation process satisfies the following inequalities

$$1+\frac{N-1}{2} \le n_k \le N .$$

- $n_k = 1+\frac{N-1}{2}$ if N is odd and there are always two *inactive* robots to become *active* at each step except at step 1 when only the *lead* robot becomes *active*.
- $n_k = N$ if only one *inactive* robot becomes *active* at each step.

After step $(k-1)$ (with $k>1$), there are i *active* robots. *Inactive* robots in the group can be reindexed in order to choose one or two appropriate robots with smallest indices, which will be *active* at step k. The implementation of step k will let one or two those robots become *active* (i.e. increase $i := i+1$ or $i := i+2$, respectively).

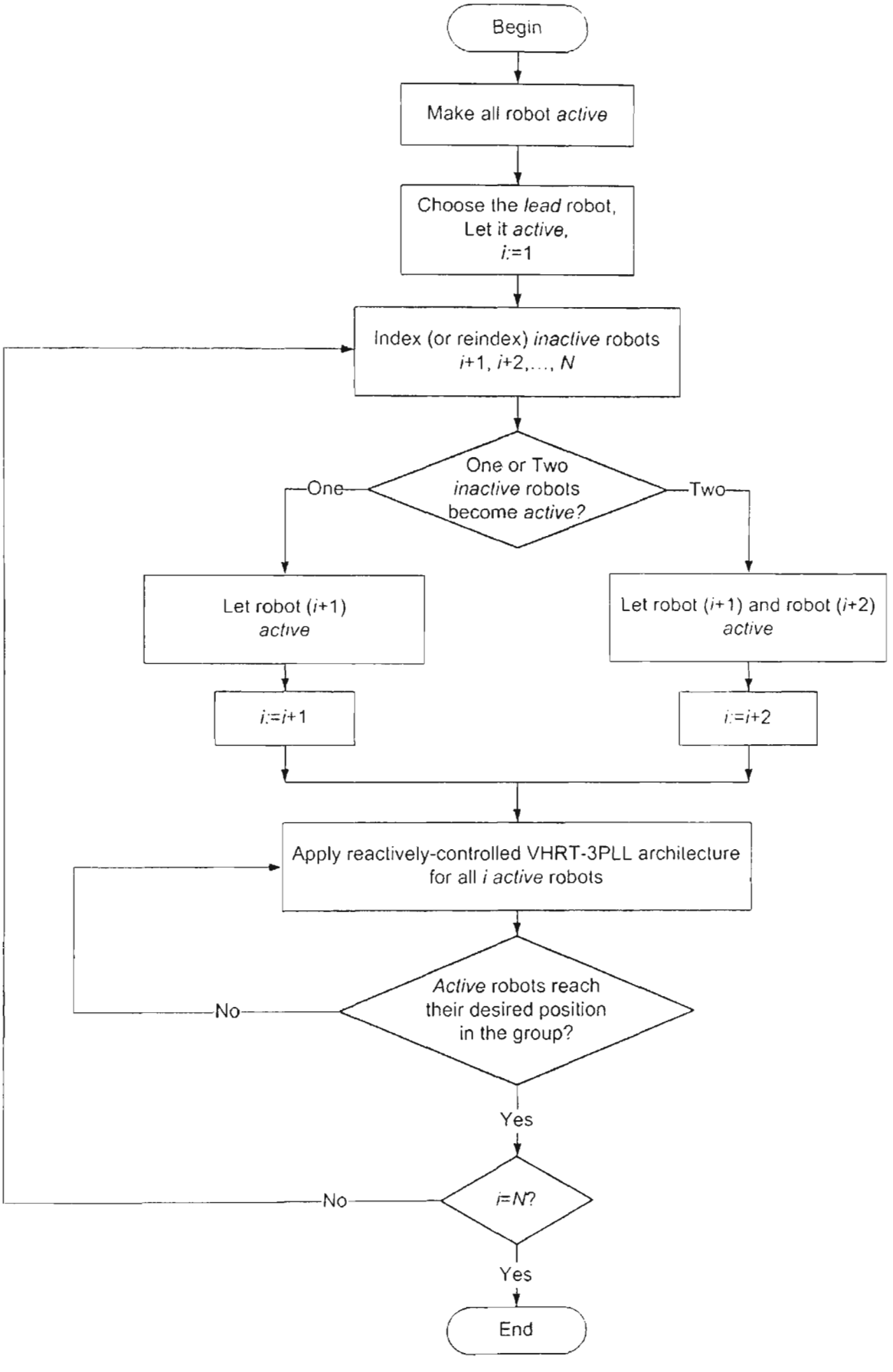


Figure 5.2: Procedure for formation initialisation for N mobile robots

Note that all robots in the group become *active* in turn from robot 1 to robot N. Thus if robot i becomes *active* at step k, robot j becomes *active* at step l and $j > i$ then $l \geq k$.

Assumption 5 ensures, at each step, the exclusion of the case when a new *active* robot (i.e. a robot becomes *active* at that step) could detect possibilities of collisions concurrently with more than one previous *active* robots. In addition, there are at most two robots become *active* at each step, therefore in the initialisation process, a robot cannot detect possibilities of collisions concurrently with more than two robots.

In practice, in order to implement the proposed procedure, there should be a centralised unit that oversees the whole group and

- chooses an appropriate shape and internal topology of formations based on the constraints in the environment and initial positions of robots
- indexes or reindexes robots if necessary, and
- decides when a step finishes and a next one begins.

The internal topology of a formation can be represented by a weighted digraph called *control graph* as discussed in the next section. All allowed control graphs can be enumerated, saved in a library and available for choosing by a centralised unit.

By applying the proposed procedure, a large group of mobile robots can be controlled to form and maintain a desired formation without inter-robot collision.

5.5 Modelling formations

A formation of N robots has one designated *lead* robot, which is indexed by 1, that directly or indirectly controls all other $(N-1)$ follower robots in the formation. Within the formation, the follower robots will be dependent on other robots for their motion. We term these robots as leaders to designate that they lead other follower robots, but distinguish them from unique *lead* robot 1. For example, using node i to represent robot i and an arrow to describe the relation from a leader to its follower, in Figure 5.3 the group of six robots has robot 1 is the *lead* robot, robots 2, 3, 4, 5 are all leaders. Note

that a robot can be a follower (tracking another robot) and simultaneously a leader (guiding another one) such as robots 2, 5 in Figure 5.3.

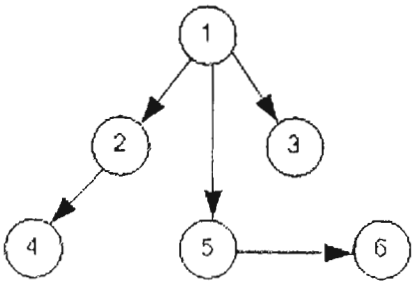


Figure 5.3: An example graph for a group of six mobile robots

5.5.1 Control graph

A formation can be made up of two components:

1. The shape variables Γ, that describe the formation internal state
2. A directed relational graph structure, the control graph or digraph, H, that represents the internal topology of the formation.

In our framework a formation could be established through the step-by-step procedure, therefore inspired by [Desai, 2002], we use a weighted digraph to model a group of mobile robots in formation. The definition of control graph is as follows.

Definition 5.1: *Control Graph*

The internal topology of a formation of N mobile robots can be represented by a weighted digraph H such that:

- *Node i represents robot i,*
- *An edge ij, which is directed from node i to node j, represents a relation between leader robot i and follower robot j,*
- *If robot i becomes active at step k and robot j becomes active at step l then edge ij is assigned a weight* $w_{ij} = l - k$.

Note that in our procedure, the *lead* robot, which is indexed by 1, becomes *active* at step 1. Robot 2 becomes *active* at step 2 and always follows the *lead* robot. Therefore, there is always an edge 12 with the weight $w_{12} = 1$ in the control graph. In addition an edge is always connected from a leader to its follower, which is becomes *active* later, thus

$$1 \le w_{ij} \le n_k - 1 \le N - 1 .$$

For example, Figure 5.4 describes a weighted digraph representing an internal topology to establish a formation for the group in Figure 5.3. Each edge is presented with its weight included. In this example, lead robot 1 becomes *active* at step 1, robot 2 and robot 3 become *active* at step 2 and follow the lead robot, robot 4 and robot 5 become *active* at step 3 (the former follows robot 2 while the later follows robot 1). Finally, robot 6 becomes *active* at step 4 and follows robot 5.

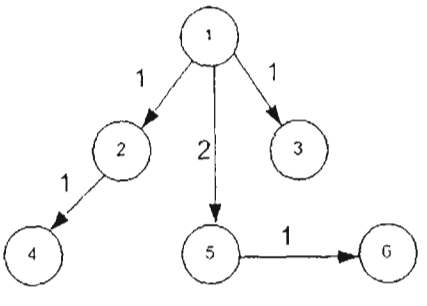


Figure 5.4: A weighted digraph for a group of six mobile robots

One possible way of representing a digraph is through an adjacency matrix. We define an adjacency matrix for a weighted digraph below.

Definition 5.2: *Adjacency matrix*

An adjacency matrix G $(N \times N)$ can be used to represent a control graph H of N mobile robots with following features.

- *Rows and columns of G are presented by nodes which represent the robots,*
- *The (i, j) element represents the weight of directed edge ij from node i to node j, i.e. $G_{ij} = w_{ij}$,*

- *Element* $G_{uv} = 0$ *if there is no edge* uv *in the graph* H.

For example, matrix G^* representing the digraph in Figure 5.4 is

$$G^* = \begin{bmatrix} 0 & 1 & 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

There are some characteristics of the control graph and its adjacency matrix representing a formation, which is established by following our step-by-step procedure.

- A follower always follows a leader with higher index. Therefore there is an edge ij only if $i < j$ and the adjacency matrix is always upper-triangular.

- The lead robot does not follow any other robots so elements in the first column of the adjacency are identically zero.

- Every robot in the group, except the lead robot, has one and only one leader. This implies that there are no two edges directing to the same node and there is one and only one element different from zero in each column, except the first column, of the adjacency matrix.

- At each step of the initialisation procedure there are at most two *inactive* robots become *active*. Thus there are at most two edges with the same weights and begin from the same node. Consequently, in the adjacency matrix there are at most two identical elements different from zero in the same row.

- If there are s rows identically zero, there are $(N\text{-}s\text{-}1)$ leaders in the group and one lead robot. For example, there are two leaders (robot 2 and robot 5) in the graph in Figure 5.4 since the third, fourth and sixth rows in G^* are identically zero.

Depending on the types of sensors used, it may be difficult for a robot to follow its leader when the leader is far way from the follower. In fact, when the sensors are of a

line-of-sight type, such as optical or acoustic sensors, the restrictions are more severe. The follower must be able to "see" the leader in the formation.

The formation can be changed, and this includes changes in the control graph and/or the shape of the formation. In those situations, it may be necessary to guarantee the "line of sight arrangement" between the leader and its followers. In general, such sensor constraints may be quite complex and are usually specific to applications.

In the following section we will prove the expression for the total number of allowable control graphs for given indexed N mobile robots. That proof implies the way to enumerate all control graphs of the group.

5.5.2 Enumeration of graphs

Consider a group of given indexed N mobile robots with lead robot 1 and $(N-1)$ follower robots.

Let $S_1(N)$ be the set of allowable control graphs of the group in which at the final step there is one robot to become *active* (i.e. robot N), $S_2(N)$ be the set of allowable control graphs of the group in which at the final step there is two robots to become *active* (i.e. robot $(N-1)$ and robot N).

Define $S(N)$ to be the set of all allowable control graph of the group. It is clear that

$$\begin{aligned} S_1(N) \cup S_2(N) &= S(N), \\ S_1(N) \cap S_2(N) &= \emptyset. \end{aligned} \tag{5.3}$$

The Cardinal numbers or numbers of elements (graphs) of $S_1(N)$, $S_2(N)$, and $S(N)$ are denoted as $Card(S_1, N)$, $Card(S_2, N)$, and $Card(S, N)$, respectively. Thus

$$Card(S, N) = Card(S_1, N) + Card(S_2, N). \tag{5.4}$$

We will state and prove the following proposition for the total number of allowable control graphs for given indexed N mobile robots.

Proposition 5.1 *The numbers of allowable control graphs of groups of* $(N>1)$ *mobile robots including the lead robot and (N-1) follower robots establish recursive sequences* $Card(S_1,N)$ *and* $Card(S_2,N)$ *with the relation:*

$$\begin{aligned} &For\ N=2: \\ &\quad Card(S_1,2)=1, \\ &\quad Card(S_2,2)=0, \\ &For\ N>2: \\ &\quad Card(S_1,N)=(N-1).\big(Card(S_1,N-1)+Card(S_2,N-1)\big), \\ &\quad Card(S_2,N)=(N-2).Card(S_1,N-1). \end{aligned} \tag{5.5}$$

Proof

For $N=2$, it is clear that there is just one possible control graph for a group of two mobile robots and this graph belongs to the set $S_1(2)$, i.e. $Card(S_1,2)=1; Card(S_2,2)=0$ (see Figure 5.5)

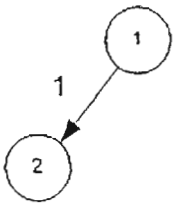


Figure 5.5: Control graph for two mobile robots

For $N>2$: Consider a group of N mobile robots.

Graphs in the set $S_1(N)$ can be created from graphs in the set $S(N-1)$ by letting robot N follow one of robots 1, 2,..,$(N-1)$ at the final step. Therefore the number of graphs in the set $S_1(N)$ is

$$Card(S_1,N)=(N-1).Card(S,N-1)=(N-1).\big(Card(S_1,N-1)+Card(S_2,N-1)\big).$$

Graphs in the set $S_2(N)$ can be created from graphs in the set $S_1(N-1)$ by letting robot N follow one of robots 1, 2,..,(*N-2*) at the final step. Note that, robots, which

become *active* at the same step, cannot be the leader- follower of each other (i.e. robot N cannot follow robot $(N-1)$). Therefore the number of graphs in the set $S_2(n)$ is

$$Card(S_2, N) = (N-2).Card(S_1, N-1).$$

■

For example, the number $Card(S, N)$ of allowable graphs for N mobile robots can be calculated as follows.

N=2:

$$Card(S_1,2) = 1,$$
$$Card(S_2,2) = 0,$$

$$Card(S,2) = Card(S_1,2) + Card(S_2,2) = 1+0 = 1.$$

N=3:

$$Card(S_1,3) = 2.Card(S,2) = 2.1 = 2,$$
$$Card(S_2,3) = 1.Card(S_1,2) = 1.1 = 1,$$

$$Card(S,3) = Card(S_1,3) + Card(S_2,3) = 2+1 = 3.$$

N=4:

$$Card(S_1,4) = 3.Card(S,3) = 3.3 = 9,$$
$$Card(S_2,4) = 2.Card(S_1,3) = 2.2 = 4,$$

$$Card(S,4) = Card(S_1,4) + Card(S_2,4) = 9+4 = 13.$$

N=5:

$$Card(S_1,5) = 4.Card(S,4) = 4.13 = 52,$$
$$Card(S_2,5) = 3.Card(S_1,4) = 3.9 = 27,$$

$$Card(S,5) = Card(S_1,5) + Card(S_2,5) = 52+27 = 79.$$

N=6:

$$Card(S_1,6) = 5.Card(S,5) = 5.79 = 395,$$
$$Card(S_2,6) = 4.Card(S_1,5) = 4.52 = 208,$$

$$Card(S,6) = Card(S_1,6) + Card(S_2,6) = 395 + 208 = 603.$$

One can obtain these patterns

N	:	2,	3,	4,	5,	6...
$Card(S_1,\bullet)$	:	1,	2,	9,	52,	395...
$Card(S_2,\bullet)$	:	0,	1,	4,	27,	208...
$Card(S,\bullet)$	:	1,	3,	13,	79,	603...

Figure 5.6 and 5.7 show all allowable control graphs for N=3 and N=4 respectively. There are 3 control graphs for a group of three mobile robots and 13 control graphs for a group of four mobile robots.

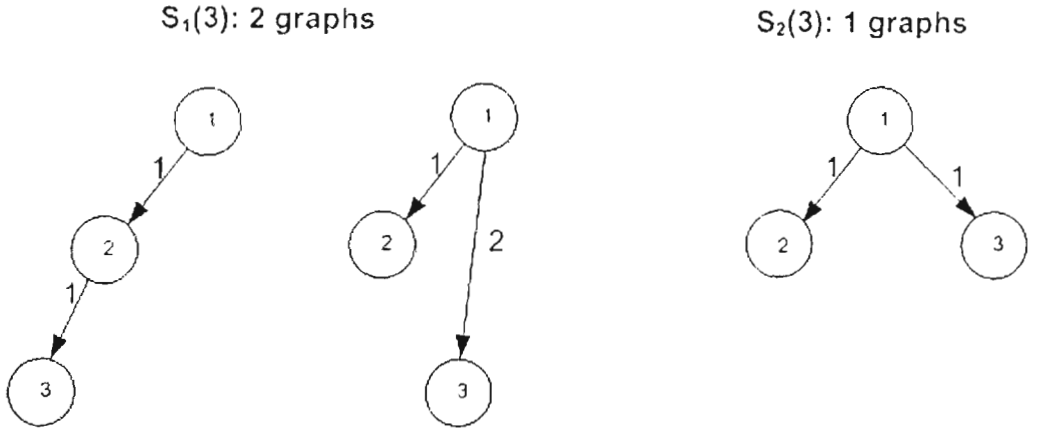


Figure 5.6: Control graphs for three mobile robots

The above proof implies the way to enumerate all allowable control graphs of a group of N mobile robots. During the process of enumerating the allowable control graphs, a library of the control graphs can be created. Each control graph could be recalled based on the constraints in the environment, positions of robots in the group and the motion of the lead robot.

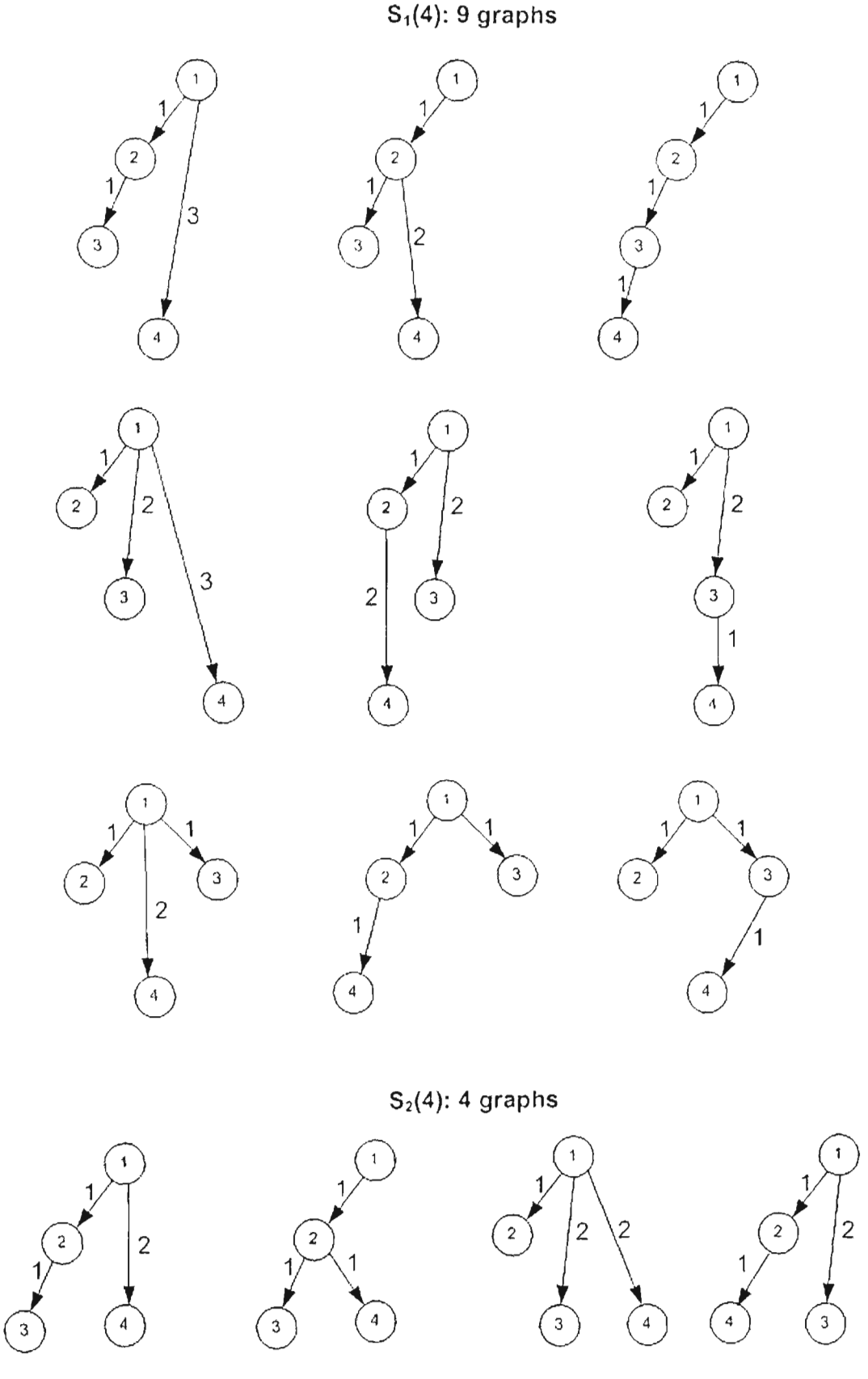


Figure 5.7: Control graphs for four mobile robots

5.6 Simulation results

We present here some simulations to illustrate the capability of using different control graphs to initialise various types of formations for a group of five mobile robots from arbitrary positions.

5.6.1 Simulation 5.1: Line formation

To illustrate the procedure proposed for a group of robots to enter a formation, we chose typically the case of five mobile robots moving to form a line formation to simulate. Parameters and conditions used were set in Table 5.1

Table 5.1: Parameters and Conditions for Simulation 5.1 and Simulation 5.2

Simulation Parameters		Robot 1	Robot 2	Robot 3	Robot 4	Robot 5
Initial Conditions	$x(0)$ (cm)	30	0	10	60	30
	$y(0)$ (cm)	0	0	50	80	-130
	$\theta(0)$ (rad)	0	0	0	2	3
	v (cm)	5				
	ω (rad/s)	0				
Parameters for desired formation	R (cm)		60	-60	60	-60
	L (cm)		0	0	0	0
Tracking margin for head robot	d (cm)		1	1	1	1
Safe distance between any two robots	$2r_{safe}$ (cm)	22	22	22	22	22
Parameters for VHRT control	λ_1 (s^{-1})		1	1	1	1
	λ_2 (s^{-1})		2	2	2	2
Parameters for *l-l* control with point A	r_A (cm)				0	0
	l_A (cm)				5	5
	T_r (s)				3	3
	δ_1 (cm)				2	2
	δ_2 (cm)				2	2

The corresponding control graph (one of 79 allowable control graphs for five mobile robots) is described in Figure 5.8

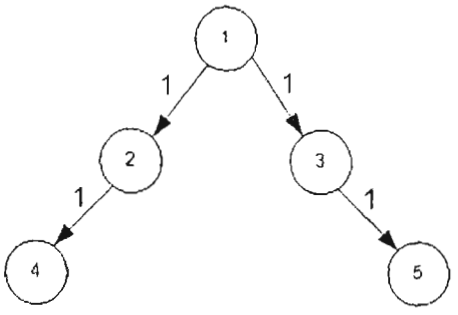


Figure 5.8: Control graph of Simulation 5.1

Following the proposed procedure, in the two first steps, three robots 1, 2, 3 formed a part of the desired formation (a "small" line). These steps took 30 seconds. At the third step, robot 4 tracked robot 2 and robot 5 tracked robot 3 to form the desired line shape.

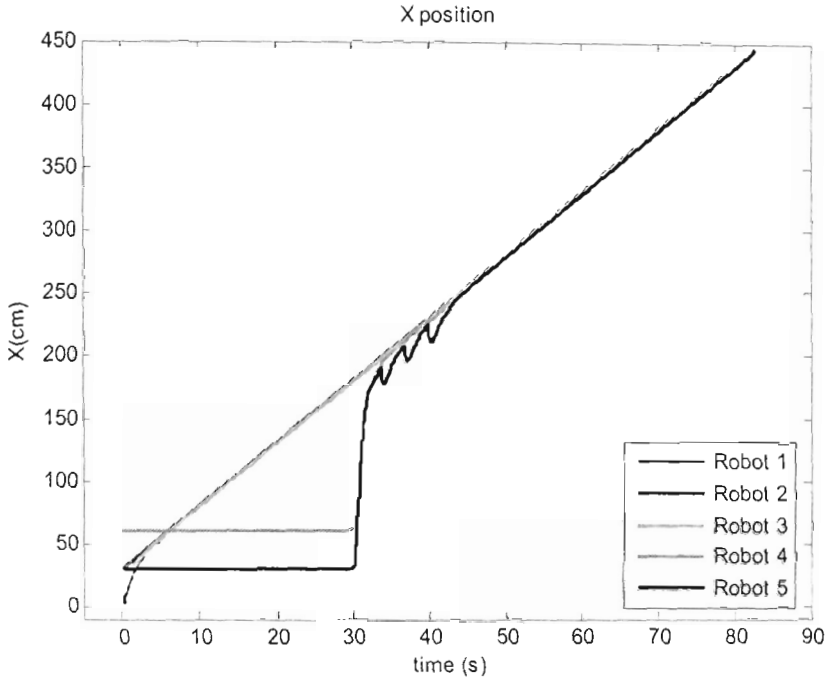
Figures 5.9 and 5.10 show the time responses of position X, Y, orientation θ and the trajectories of five mobile robots in the global coordinates. It is observed that robot 5 could possibly collide with robot 4 at $t = 30.52s$. After avoiding collision by using the proposed reactive control scheme 1, robot 5 again could possibly collide with robot 2 at $t = 33.56s$, with robot 1 at $t = 36.64s$, and with robot 3 at $t = 39.71s$. At those instances, robot 5 also used proposed reactive control scheme 1 to avoid collision and then switched back to tracking control to eventually establish the desired formation.

Applying reactive control schemes for collision avoidance, there may be abrupt changes in velocities of the follower (i.e. robot 3 of concerning robots). As mentioned in the previous chapter, the choices of l_{13}^d, l_{23}^d and T_r should be carefully concerned to alleviate the sudden changes of velocities of the follower when switching from VHRT to 3PLL control.

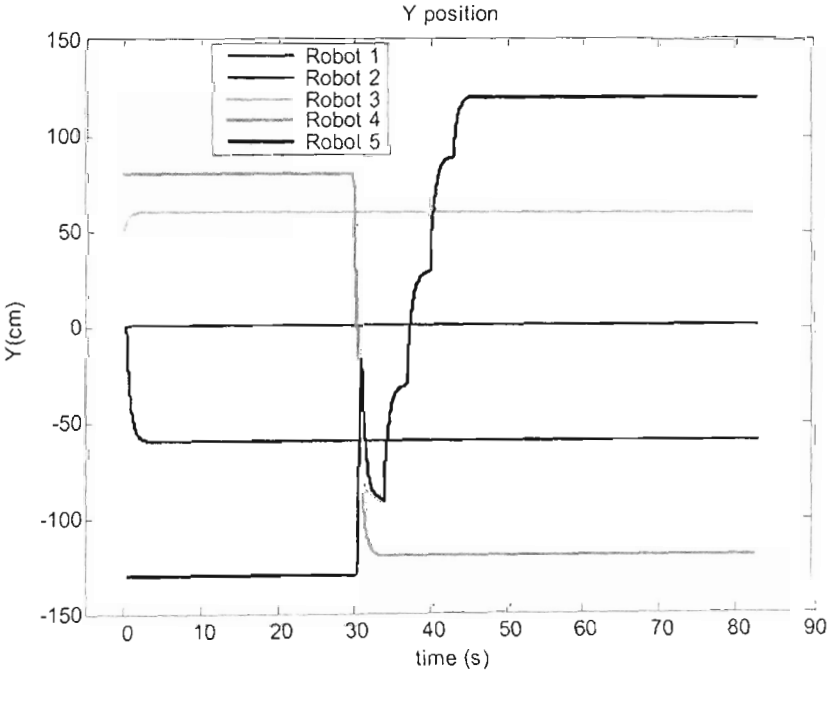
This simulation demonstrates the capability of avoiding collision and forming a desired formation for a group of multiple mobile robots.

5.6.2 Simulation 5.2: Line formation – No collision

This simulation used the same parameters and conditions as in the Simulation 5.1 (see Table 5.1) but with a different control graph. The control graph used for this simulation is described in Figure 5.11.



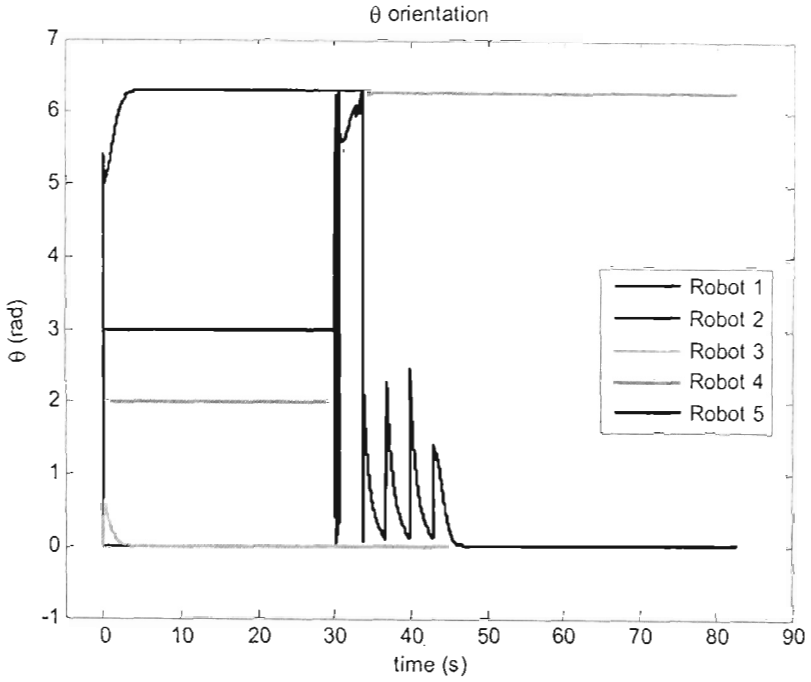
a)



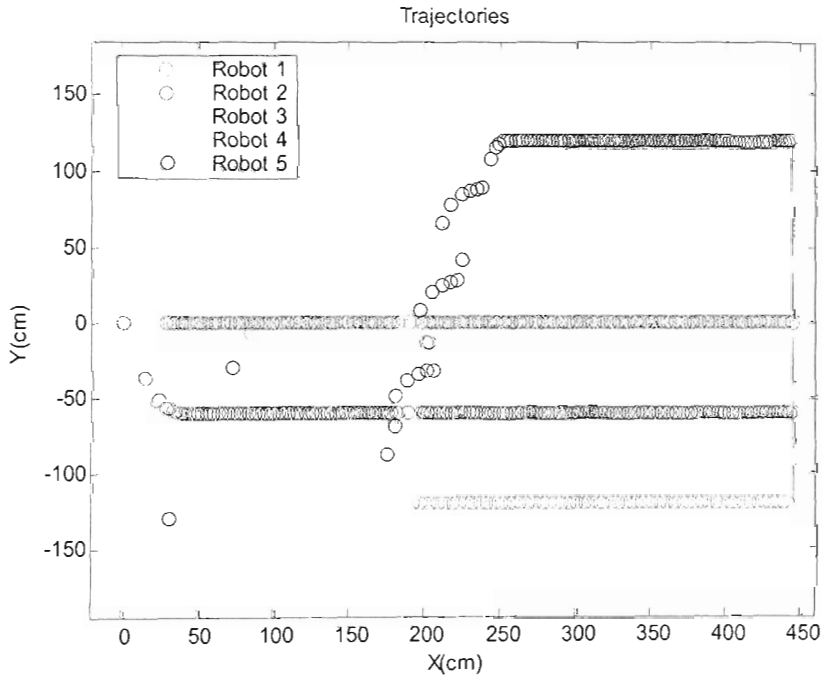
b)

Figure 5.9: Simulation 5.1 results - X, Y position

a) X position b) Y position



a)



b)

Figure 5.10: Simulation 5.1 results – θ orientation and trajectories

a) θ- orientation b) Trajectories

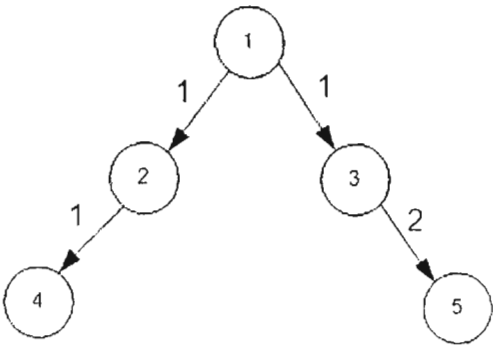


Figure 5.11: Control graph of Simulation 5.2

As in previous simulation, in the two first steps, three robots 1, 2, 3 formed a part of the desired formation (a "small" line). These steps took 30 seconds. At the third step, only robot 4 became *active* and tracked robot 2. This step took 10s seconds. Finally, at the fourth step, robot 5 became *active* and tracked robot 3 to form the desired line formation.

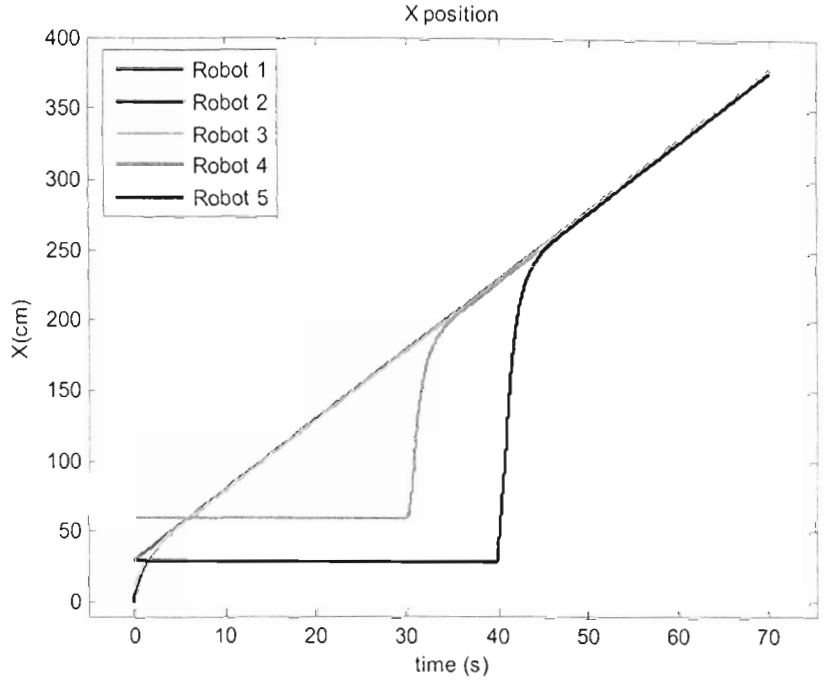
Figures 5.12 and 5.13 show the time responses of position X, Y, orientation θ and the trajectories of five mobile robots in the global coordinates. By letting robot 5 become *active* after robot 4 does, there is no possibility of collisions in this simulation.

This simulation illustrates that an appropriate choice of the control graph can lessen possibilities of collisions among robots in the group.

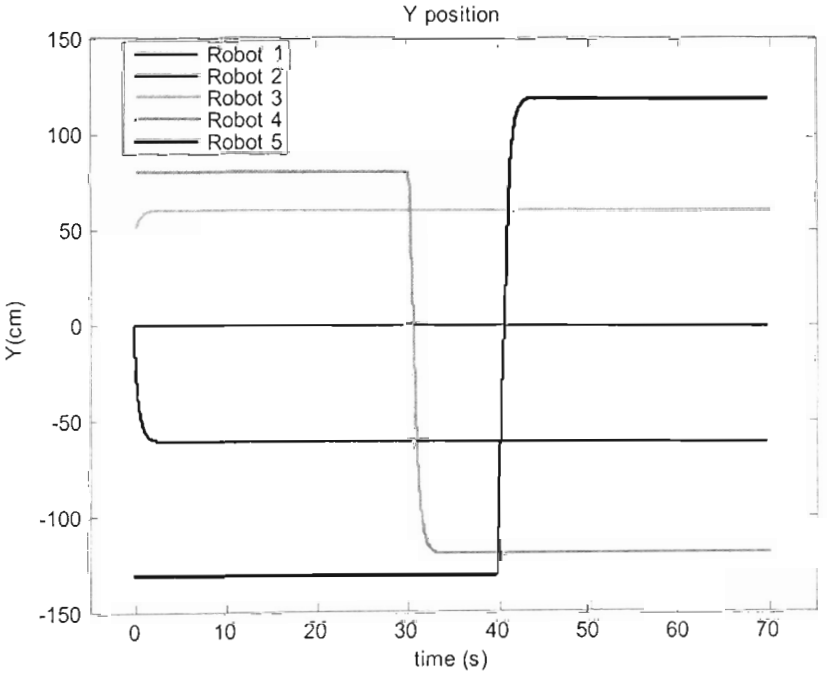
5.6.3 Simulation 5.3: Diamond-like formation

We considered the case of five robot moving to form a diamond-like formation. Parameters and conditions used were set in Table 5.2 with the control graph described in Figure 5.14.

Three robots 1, 2, 3 formed a part of the desired formation (a wedge) in the first two steps by following the proposed procedure. After that, at $t = 30s$, the third step began with robot 4 and robot 5 tracking robot 1 to form the desired diamond shape.



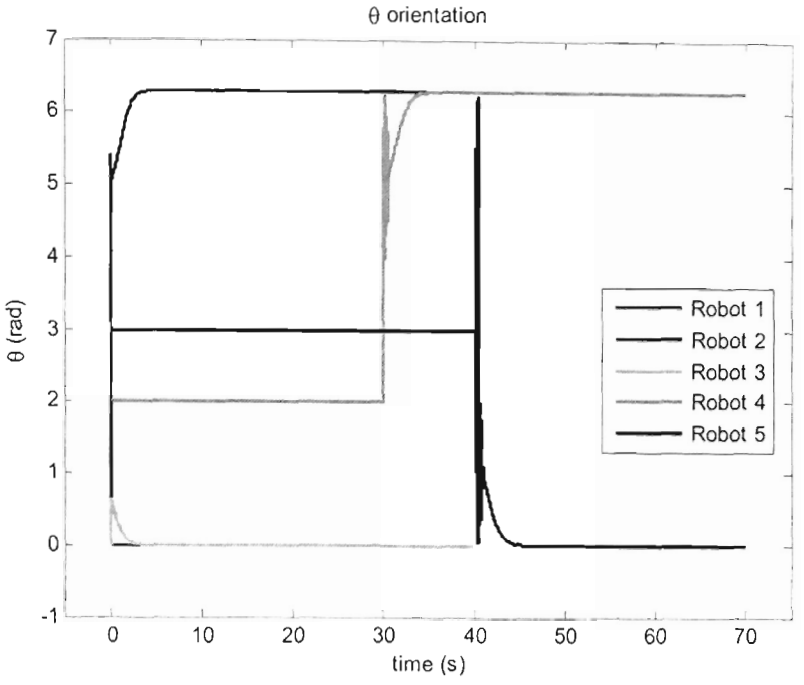
a)



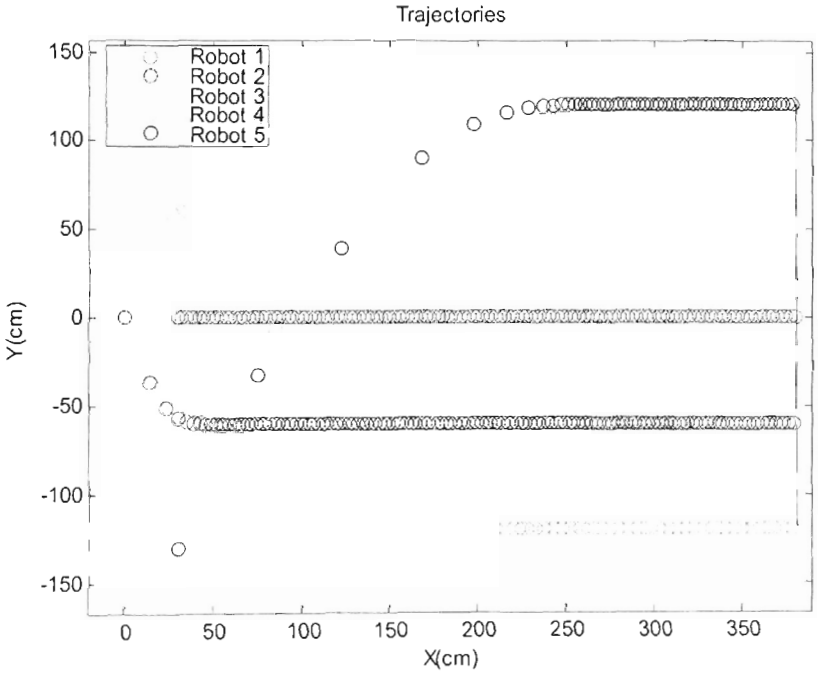
b)

Figure 5.12: Simulation 5.2 results - X, Y position

a) X position b) Y position



a)



b)

Figure 5.13: Simulation 5.2 results – θ orientation and trajectories

a) θ- orientation b) Trajectories

Table 5.2: Parameters and Conditions for Simulation 5.3

Simulation Parameters		Robot 1	Robot 2	Robot 3	Robot 4	Robot 5
Initial Conditions	$x(0)$ (cm)	30	0	10	150	150
	$y(0)$ (cm)	0	0	50	180	-150
	$\theta(0)$ (rad)	0	0	0	2	$\pi/2$
	v (cm)	5				
	ω (rad/s)	0				
Parameters for desired formation	R (cm)		85	-85	0	0
	L (cm)		40	40	40	80
Tracking margin for head robot	d (cm)		1	1	1	1
Safe distance between any two robots	$2r_{safe}$ (cm)	20	20	20	20	20
Parameters for VHRT control	λ_1 (s^{-1})		1	1	1	1
	λ_2 (s^{-1})		2	2	2	2
Parameters for *l-l* control with point A	r_A (cm)				0	0
	l_A (cm)				5	5
	T_r (s)				3	3
	δ_1 (cm)				2	2
	δ_2 (cm)				2	2

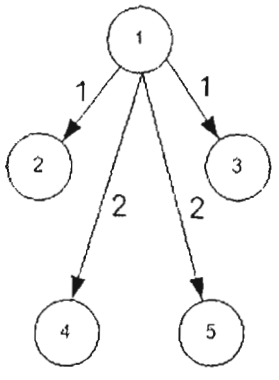


Figure 5.14: Control graph of Simulation 5.3 and Simulation 5.4

Figures 5.15 and 5.16 show the time responses of position X, Y, orientation θ and the trajectories of five mobile robots in the global coordinates. It is observed robot 5 could possibly collide with robot 2 at $t = 30.22s$, and robot 4 with robot 3 at $t = 30.45s$. After avoiding collision by using the proposed reactive control schemes, robot 4 and robot 5 could switch back to tracking control to eventually establish the desired formation.

Our simulation results show the validity of our proposed framework to initialise and establish a desired formation for a group of mobile robots while ensuring inter-robot collision avoidance.

5.6.4 Simulation 5.4: Wedge formation

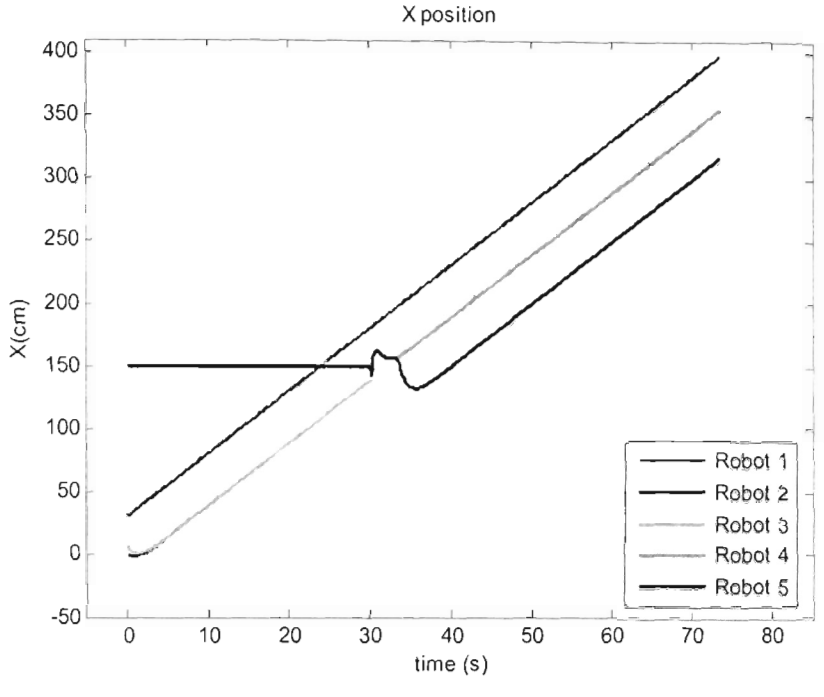
In this simulation, we chose five robots to form a wedge formation. The control graph of this simulation is the same as the one of Simulation 5.3 and is described in Figure 5.14. Parameters and conditions of robots were set in Table 5.3

Table 5.3: Parameters and Conditions for Simulation 5.4

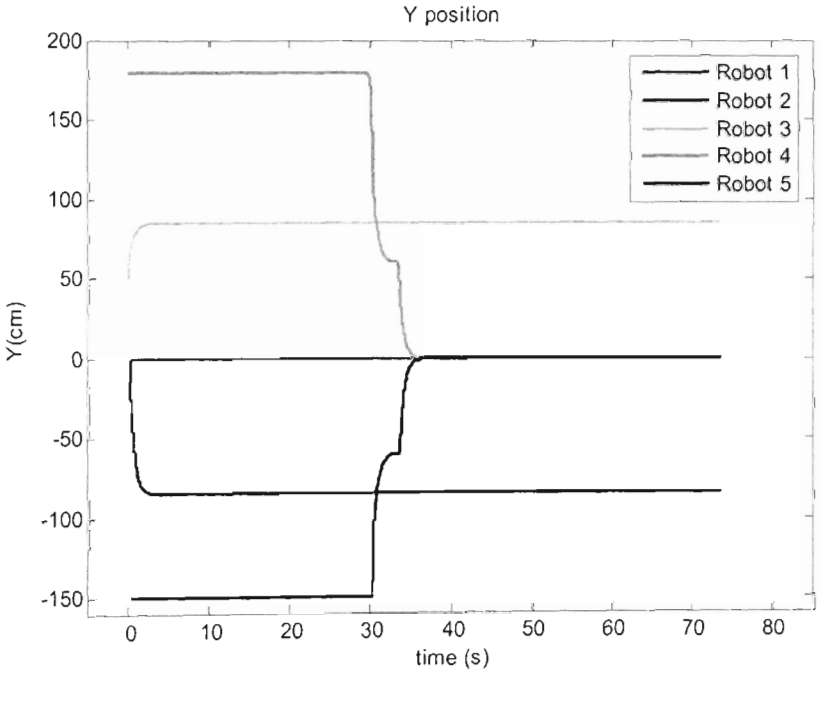
Simulation Parameters		Robot 1	Robot 2	Robot 3	Robot 4	Robot 5
Initial Conditions	$x(0)$ (cm)	30	0	10	50	330
	$y(0)$ (cm)	0	0	50	80	-140
	$\theta(0)$ (rad)	0	0	0	2	3
	v (cm)	5				
	ω (rad/s)	0				
Parameters for desired formation	R (cm)		50	-50	50	-50
	L (cm)		50	50	100	100
Tracking margin for head robot	d (cm)		1	1	1	1
Safe distance between any two robots	$2r_{safe}$ (cm)	22	22	22	22	22
Parameters for VHRT control	λ_1 (s^{-1})		1	1	1	1
	λ_2 (s^{-1})		2	2	2	2
Parameters for l-l control with point A	r_A (cm)				0	0
	l_A (cm)				5	5
	T_r (s)				3	3
	δ_1 (cm)				2	2
	δ_2 (cm)				2	2

After first two steps of the initialisation procedure as in the Simulation 5.3 to form a small wedge, at $t = 30s$, robot 4 and robot 5 began to track robot 1 to join the group.

Figures 5.17 and 5.18 show the time responses of position X, Y, orientation θ and the trajectories of five mobile robots in the global coordinates.



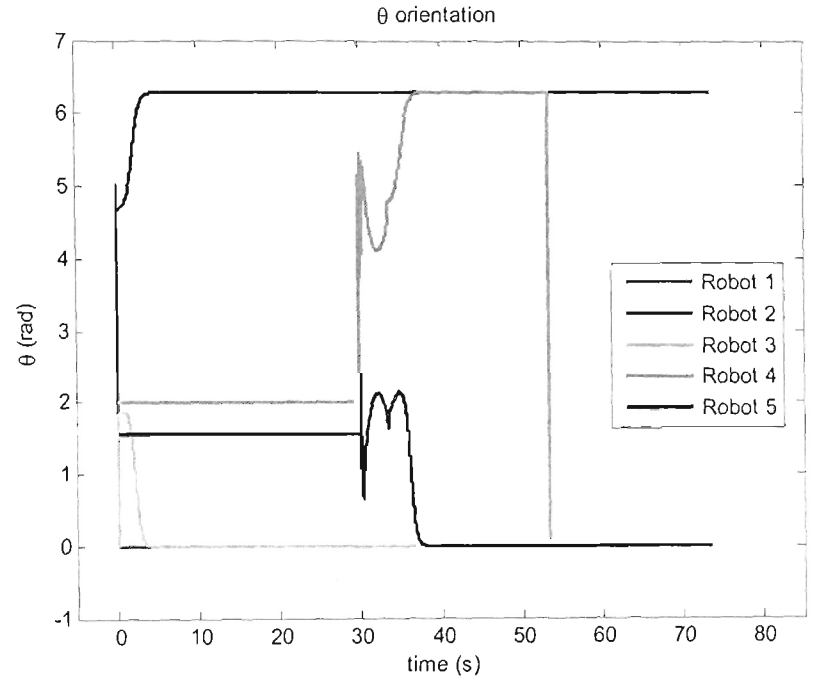
a)



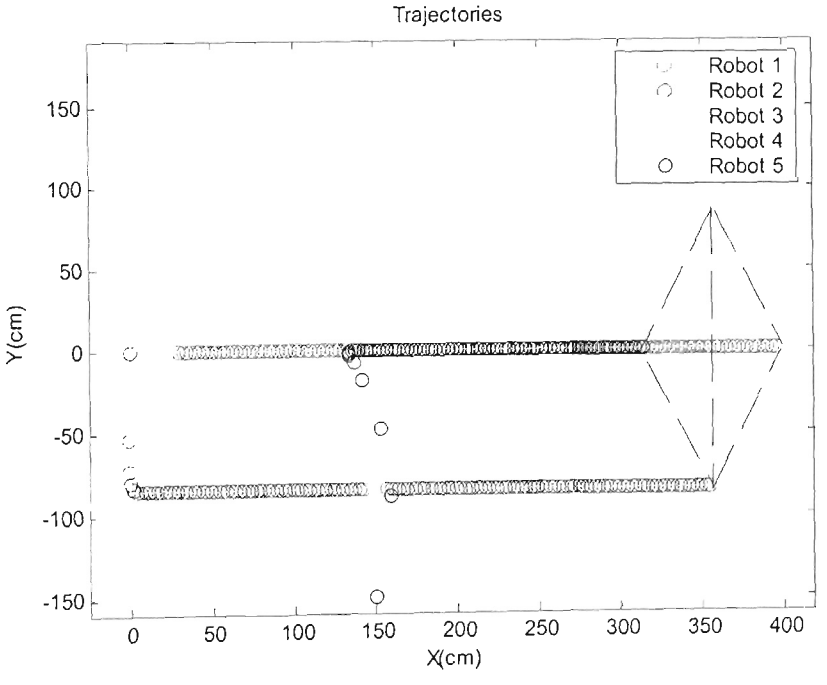
b)

Figure 5.15: Simulation 5.3 results - X, Y position

a) X position b) Y position



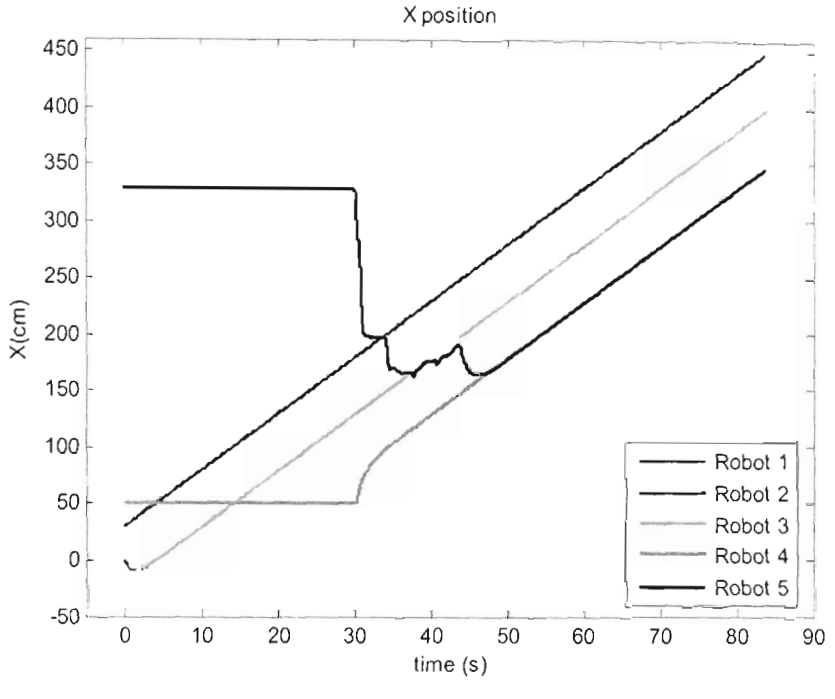
a)



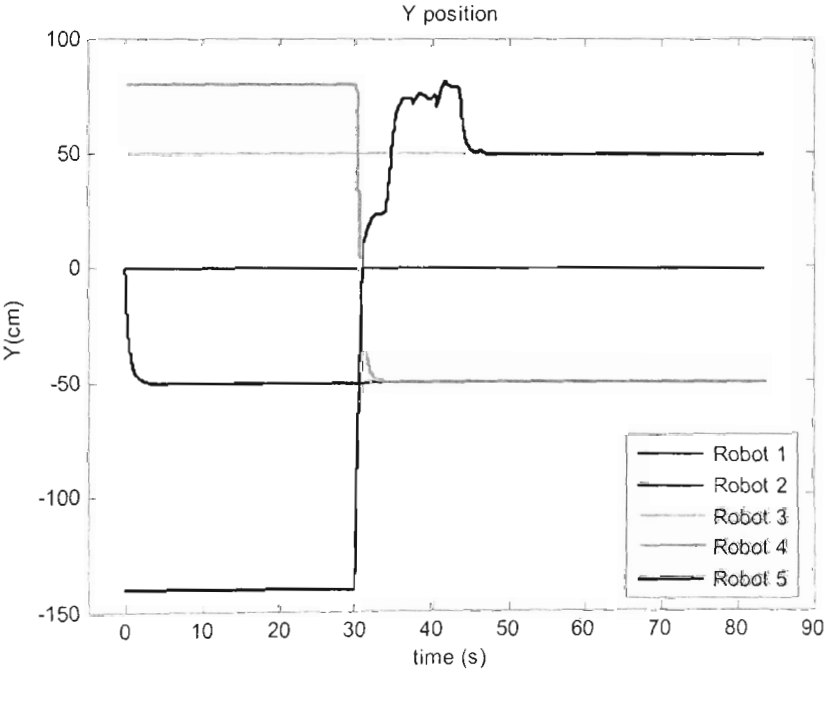
b)

Figure 5.16: Simulation 5.3 results – θ orientation and trajectories

a) θ- orientation b) Trajectories



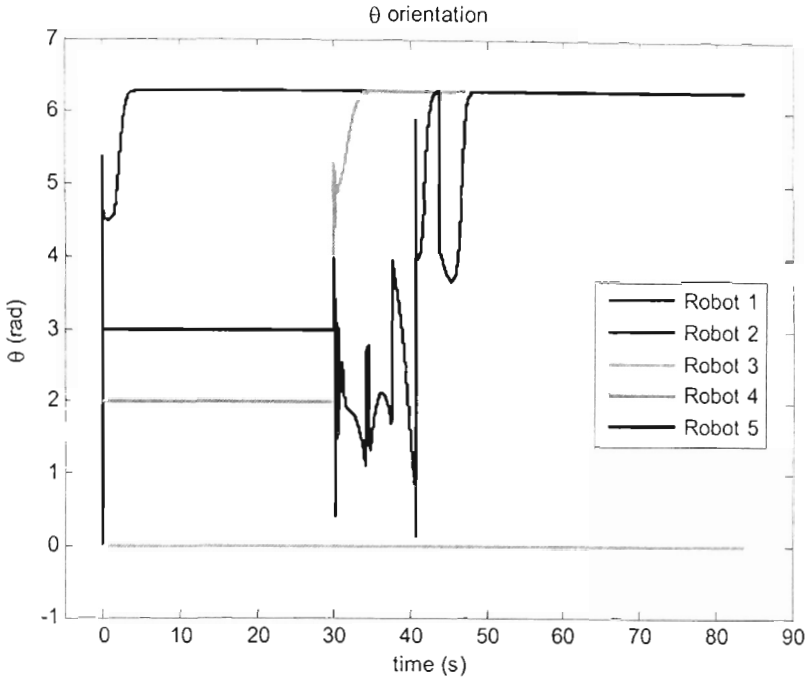
a)



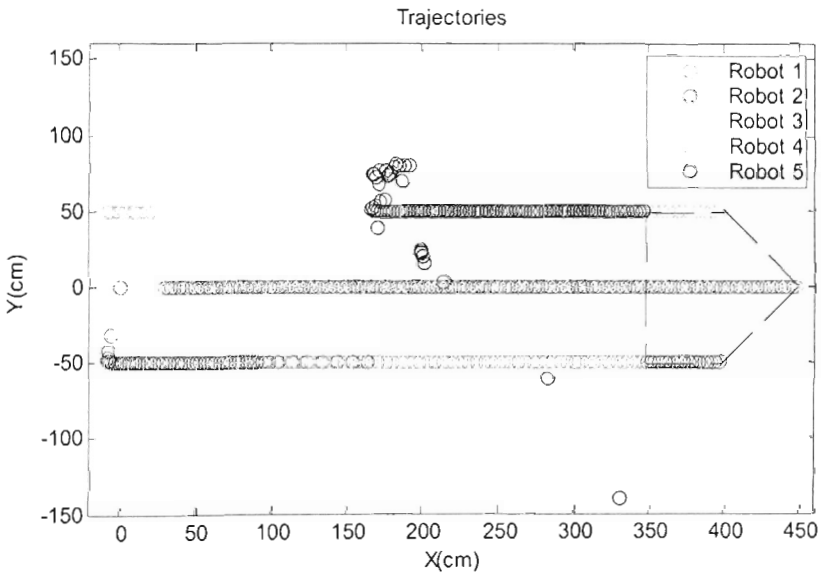
b)

Figure 5.17: Simulation 5.4 results - X, Y position

a) X position b) Y position



a)



b)

Figure 5.18: Simulation 5.4 results – θ orientation and trajectories

a) θ- orientation b) Trajectories

It is observed the possibility of collision between robots 5 and robot 1 at $t = 31.08s$. At that moment reactive control scheme 1 was used to avoid the collision. However, when robot 5 switched back to tracking control, the possibilities of collision between it and robot 3 could be observed again at $t = 34.47s, 37.56s$ and 40.7s. Therefore robot 5 had to switch between tracking and collision avoiding controls several times before it can reach to the desired position in the group's formation.

This simulation demonstrates that in some cases the reactive control schemes have to be applied several times to avoid collisions among robots

5.7 Conclusion

This chapter has presented a framework for initialization formation for a group of N mobile robots. The proposed framework includes a step-by-step procedure incorporated with some reactive control schemes based on VHRT and 3PLL control laws. Weighted digraphs are used to model the formations established by the step-by-step procedure. We also have stated and proven a proposition, which is useful for enumerating all allowable control graphs for a group of given indexed N mobile robots. Under some assumptions as in the case for a group of three mobile robots together with the new one of desired formations, a group of multiple mobile robots may be initialized and established desired formations without inter-robot collisions by using this framework.

Chapter 6

Observer-based decentralised approach to robotic formation

6.1 Introduction

Control of a group of mobile robots in a formation requires not only environmental sensing but also communication among vehicles. Enlarging the size of the platoon of vehicles causes difficulties due to communications bandwidth limitations. Decentralized control may be an appropriate approach in those cases when the states of all vehicles cannot be obtained in a centralised manner.

In [Beard *et al.*, 2001], a control architecture for formation flying is proposed using formation and supervisor units in a centralized manner, and local controllers are designed to estimate the states of the local instantiations of these units. However, interconnections between formation and the local agents, interactions between the supervisor unit and behaviours are not introduced, and also the observer design mentioned in local control is not detailed. A framework for decentralized control of autonomous vehicles is proposed in [Stilwell and Bishop, 2000], using nonlinear observers to estimate the complete system state with minimal explicit communications between agents. The examples therein illustrate an autonomous platoon with a very

simple model for the vehicle dynamics. A specific communication network topology is examined later in [Stilwell, 2002], where platoon-level functions representing global features that can be measured by an exogenous system. To implement these results in a realistic setting, a separate controller would be designed for each of a series of trajectories, and then the controllers would be gain scheduled as the vehicles move along the trajectories.

In moving toward a suitable architecture for multi-agent system control, this chapter, motivated by [Stilwell, 2002] and [Ha and Trinh, 2004], is devoted to the decentralized implementation of a global state-feedback controller for a platoon of mobile robots in a formation under a decentralized information structure. The multi-agent system comprises generally N robots, each with a local control station. The control input for the ith station is calculated from the information contained in its local input and output signals only. Decentralised observers are also proposed here but unlike the approach by [Stilwell and Bishop, 2000], no explicit flow of information takes place among the control stations. It should be pointed out that nonlinearity, under-actuation and complexity of the formation model would make it difficult to implement any decentralised control strategies for a large number of robots unless they have been properly initialised. It is expected therefore that the robots formation motion mentioned in this Chapter has passed an initialisation phase by using a technique described previously in Chapters 3-5, and also the formation trajectories are linear.

This chapter is organised as follows. After the introduction, Section 6.2 presents the system description and formulates the problem. The main development of the proposed approach is detailed in Section 6.3. The design procedure is illustrated in Section 6.4 with simulation and experimental results included for a group of mobile robots. The conclusion is given in Section 6.5.

6.2 Modelling

6.2.1 Model of a nonholonomic mobile robot

A mobile robot can be described by a common kinematic model as:

$$\begin{aligned} \dot{x} &= v\cos\theta, \\ \dot{y} &= v\sin\theta, \\ \dot{\theta} &= \omega, \end{aligned} \tag{6.1}$$

where (x, y) is the centre point on the wheel axis, $\theta \in R$ is the orientation and inputs v and ω are the translational and angular velocities respectively.

By linearising around a specific trajectory with $v = a = const$ and $\theta = b = const$ the corresponding velocities are $\dot{x} = a\cos b$, $\dot{y} = a\sin b$, and $\dot{\theta} = 0$. By selecting new variables

$$\begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{\theta} \end{bmatrix} = \begin{bmatrix} x - (a\cos b).t \\ y - (a\sin b).t \\ \theta - b \end{bmatrix}, \quad \begin{bmatrix} \bar{v} \\ \bar{\omega} \end{bmatrix} = \begin{bmatrix} v - a \\ \omega \end{bmatrix}, \tag{6.2}$$

one can obtain a linear time-invariant system for the robot as:

$$\begin{bmatrix} \dot{\bar{x}} \\ \dot{\bar{y}} \\ \dot{\bar{\theta}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -a\sin b \\ 0 & 0 & a\cos b \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{\theta} \end{bmatrix} + \begin{bmatrix} \cos b & 0 \\ \sin b & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{v} \\ \bar{\omega} \end{bmatrix}. \tag{6.3}$$

Note that this linearised model is valid when the motion of a robot in formation is near a specified trajectory [Stilwell, 2002].

6.2.2 Modelling a group of mobile robots in a formation

Consider a system composed of N mobile robots, uncoupled and modeled by (6.1). Each robot has a local controller that generates the local control signals based on local measured signals and signals broadcast exogenously.

The states of the whole system can be described by:

$$X = \begin{bmatrix} x_\Sigma & y_\Sigma & \theta_\Sigma \end{bmatrix}^T, \tag{6.4}$$

where $x_\Sigma = \begin{bmatrix} x_1 & x_2 \dots x_N \end{bmatrix}^T$, $y_\Sigma = \begin{bmatrix} y_1 & y_2 \dots y_N \end{bmatrix}^T$, $\theta_\Sigma = \begin{bmatrix} \theta_1 & \theta_2 \dots \theta_N \end{bmatrix}^T$, and where x_i, y_i, θ_i are positions and orientation of the i-th robot. The control input is $u = \begin{bmatrix} v_\Sigma & \omega_\Sigma \end{bmatrix}^T$, where

$v_\Sigma = [v_1\ v_2 \ldots v_N]^T$ and $\omega_\Sigma = [\omega_1\ \omega_2 \ldots \omega_N]^T$, with v_i, ω_i being respectively the translational and angular velocities of the i-th robot, $i = 1,2,\ldots,N$.

Globally, there are features of the platoon that can be measured exogenously. These features such as the vehicle average position are referred to as action reference [Kang *et al.*, 2000], dynamic coordination variables [Beard *et al.*, 2001], or platoon-level functions [Stilwell, 2002]. In this chapter, they are denoted $h(X)$, a function of the entire platoon state, assumed to be linear and differentiable, broadcast to all robots. In the control of a group of N mobile robots in a formation the state variables and platoon-level functions characterize the state of the overall system with respect to a global objective (i.e. getting into and maintaining a formation pattern). Thus, the model of the platoon can be written as:

$$\dot{S}(t) = AS(t) + Bu(t), \tag{6.5a}$$

$$y(t) = CS(t), \tag{6.5b}$$

where $S(t) = [\overline{X}\ \ h]^T \in R^{n_\Sigma}$ is the state vector, which is the augmentation of the global system state variables under consideration $\overline{X}$ with the platoon level functions h, $u(t) \in R^{m_\Sigma}$ and $y(t) \in R^{r_\Sigma}$ are the input and output vectors, respectively. Under the linearised conditions, matrices $A \in R^{n_\Sigma \times n_\Sigma}$, $B \in R^{n_\Sigma \times m_\Sigma}$ and $C \in R^{r_\Sigma \times n_\Sigma}$ are real constant.

Centralized control can be implemented if full information of S is made available to individual robots from a central unit. It becomes however very difficult when the size of the system is quite large. Following the approach proposed in [Stilwell, 2002], where the platoon-level functions representing integrated error signals are broadcast from an exogenous system, an alternative technique to the robotic formation control problem is proposed in this chapter by using observer-based decentralized controllers (see Figure 6.1).

6.3 Observer-based decentralised control

Consider a linear time-invariant multivariable system described by (6.5). Without loss of generality, it is assumed that the triplet (A,B,C) is controllable and observable. Let N denotes the number of local control stations for N robots of the platoon.

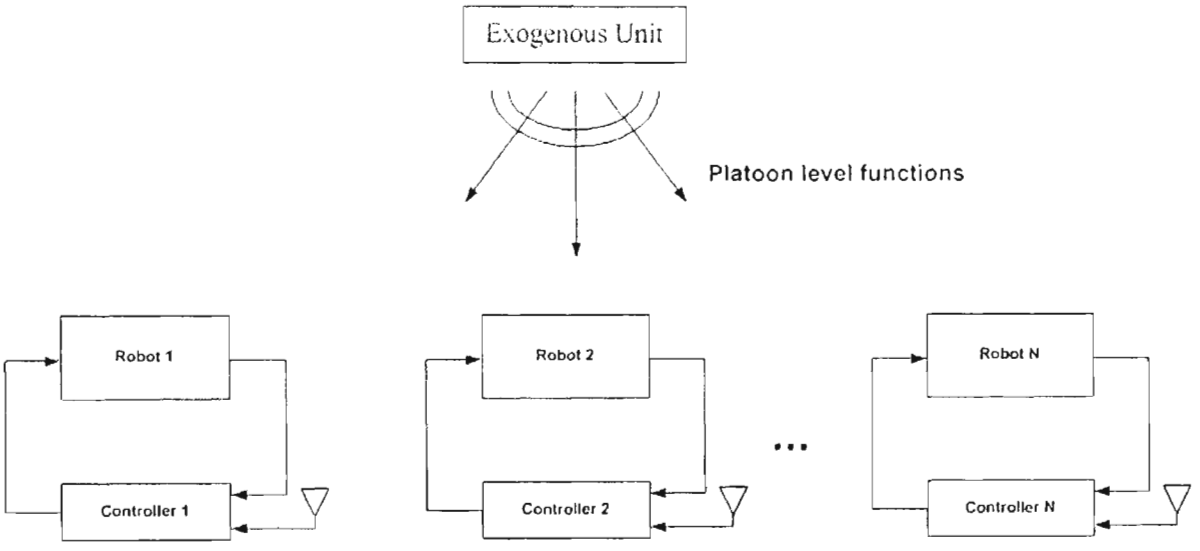


Figure 6.1: Decentralised control model

Let the elements of the input vector $u(t)$ and output vector $y(t)$ be arranged so that

$$u(t) = [u_1^T(t),\ u_2^T(t),\ldots,\ u_N^T(t)]^T, \tag{6.6a}$$

$$y(t) = [y_1^T(t),\ y_2^T(t),\ldots,\ y_N^T(t)]^T, \tag{6.6b}$$

where $u_i(t) \in R^{m_i}$ and $y_i(t) \in R^{r_i}$ $(i = 1,2,\ldots,N)$ are respectively the input and output vectors of the i-th agent (e.g., $u_i = [v_i\ \omega_i]^T$). Accordingly, the system (6.5) can be rewritten as

$$\dot{S}(t) = AS(t) + \sum_{i=1}^{N} B_i u_i(t), \tag{6.7a}$$

$$y_i(t) = C_i S(t),\quad i = 1,2,\ldots,N, \tag{6.7b}$$

where $B_i \in R^{n_\Sigma \times m_i}$ and $C_i \in R^{r_i \times n_\Sigma}$ are respectively sub-matrices of B and C, determined according to equations (6.6).

6.3.1 Assumptions

Let us first introduce some assumptions.

- **Assumption 1**: The global system (A, B, C) is controllable and observable.
- **Assumption 2**: There exist no decentralised fixed modes [Wang and Davison, 1973] associated with triplets (B_i, A, C_i), or if existing, they are assumed to be stable.
- **Assumption 3**: Information available to the ith control station, $\Im_i(t)$ includes only the local output and control of the ith station:

$$\Im_i(t) = \{y_i(t), u_i(t)\},\ i = 1,2,...,N. \tag{6.8}$$

- **Assumption 4**: A satisfactory global state feedback control law has been found of the form

$$u(t) = FS(t), \tag{6.9}$$

where $F \in R^{m_\Sigma \times n_\Sigma}$, by using any standard state feedback control method to obtain the satisfaction of some system performance index.

- **Assumption 5**: Incidents for obstacle and inter-robot collision have been avoided.

6.3.2 Problem statement

Taking into account the constraint of the decentralised information structure (6.8), the objective here is to design decentralised controllers of the form

$$u_i(t) = f_i\{\Im_i(t), t\},\ i = 1,2,...,N, \tag{6.10}$$

using only information available to local control stations, i.e. $\Im_i(t)$ such that the multi-robot system (6.7) is stable with satisfactory performance as prescribed in the global

control law (6.9). To achieve the control objective the global control (6.9) will be constructed dynamically via decentralised linear functional observers that receive only $\Im_i(t)$ as their inputs.

Let the global controller (6.9) be partitioned as

$$u_i(t) = F_i S(t)\,;\ i = 1,2,...,N, \tag{6.11}$$

where $F_i \in R^{m_i \times n_\Sigma}$. The decentralised controllers (6.10) are proposed to have the observer-based form:

$$u_i(t) = (K_i L_i + W_i C_i)S(t) = K_i z_i(t) + W_i y_i(t), \tag{6.12a}$$

$$\dot{z}_i(t) = E_i z_i(t) + L_i B_i u_i(t) + G_i y_i(t), \quad i = 1,2,...,N, \tag{6.12b}$$

where $F_i = K_i L_i + W_i C_i$, $z_i = L_i S(t) \in R^{p_i}$ is the state vector of system (6.12); and real constant matrices $K_i \in R^{m_i \times p_i}$, $L_i \in R^{p_i \times n_\Sigma}$, $W_i \in R^{m_i \times r_i}$, $E_i \in R^{p_i \times p_i}$, and $G_i \in R^{p_i \times r_i}$ are to be determined.

6.3.3 Observer development

Let us assume, without loss of generality, that matrix C_i has full row rank, i.e. $rank(C_i) = r_i$, and takes the following canonical form

$$C_i = [I_{r_i} \quad 0], \tag{6.13}$$

where I_{r_i} is an identity matrix of dimension r_i.

Let the global control input matrix B be partitioned as

$$B = [B_i \quad B_{r_i}], \tag{6.14}$$

where $B_{r_i} \in R^{n_\Sigma \times (m_\Sigma - m_i)}$. Accordingly, (6.5) can be expressed as

$$\dot{S}(t) = AS(t) + B_i u_i(t) + B_{r_i} u_{r_i}(t), \tag{6.15}$$

$$y_i(t) = C_i S(t), \quad i = 1,2,...,N, \tag{6.16}$$

where $u_{r_i}(t)$ contains $(N-1)$ input vectors of the remaining $(N-1)$ control stations from other robots in the system.

Let an error vector $e_i(t)$ be defined as

$$e_i(t) = z_i(t) - L_i S(t); \; i = 1,2,...,N. \tag{6.17}$$

By some simple manipulations, the following error equation is obtained

$$\begin{aligned} \dot{e}_i(t) &= \dot{z}_i(t) - L_i \dot{S}(t) = E_i z_i(t) + L_i B_{r_i} u_{r_i}(t) + G_i y_i(t) - L_i AS(t) - L_i B_i u_i(t) - L_i B_{r_i} u_{r_i}(t) \\ &= E_i e_i(t) + (G_i C_i - L_i A + E_i L_i) S(t) - L_i B_{r_i} u_{r_i}(t). \end{aligned} \tag{6.18}$$

Therefore, (6.12b) can act as a decentralized linear functional observer for system (6.15-6.16), provided that matrix E_i is chosen to be asymptotically stable and matrices G_i and L_i fulfill the following constraints

$$G_i C_i - L_i A + E_i L_i = 0, \tag{6.19}$$

$$L_i B_{r_i} = 0, \tag{6.20}$$

$$F_i = K_i L_i + W_i C_i. \tag{6.21}$$

Matrix E_i can be chosen according to the desired dynamics of the observer to be constructed. There are thus four unknown matrices (G_i, L_i, K_i and W_i) in equations (6.19)-(6.21) to be solved for.

Using (6.13), equations (6.19) and (6.21) can be expressed as

$$G_i = (L_i A - E_i L_i) \begin{bmatrix} I_{r_i} \\ 0 \end{bmatrix}, \tag{6.22a}$$

$$(L_i A - E_i L_i)\begin{bmatrix} 0 \\ I_{(n_\Sigma - r_i)} \end{bmatrix} = 0, \tag{6.22b}$$

and

$$W_i = (F_i - K_i L_i)\begin{bmatrix} I_{r_i} \\ 0 \end{bmatrix}, \tag{6.23a}$$

$$(F_i - K_i L_i)\begin{bmatrix} 0 \\ I_{(n_\Sigma - r_i)} \end{bmatrix} = 0. \tag{6.23b}$$

It is clear from equations (6.22a) and (6.23a) that matrices G_i and W_i can be directly derived, once matrices K_i and L_i are obtained. It remains therefore to solve equations (6.20), (6.22b), and (6.23b) for matrices K_i and L_i.

Let matrices F_i and L_i be partitioned as follows

$$F_i = [f_1 \quad f_2 \quad \cdots \quad f_{r_i} \mid f_{r_i+1} \quad f_{r_i+2} \quad \cdots \quad f_{n_\Sigma}], \tag{6.24}$$

and

$$L_i = [\, l_1 \quad l_2 \quad \cdots \quad l_{r_i} \mid l_{r_i+1} \quad l_{r_i+2} \quad \cdots \quad l_{n_\Sigma}], \tag{6.25}$$

where $f_j = [f_{1,j}\ f_{2,j} \cdots f_{m_i,j}]^T \in R^{m_i}, l_j = [l_{1,j}\ l_{2,j} \cdots l_{p_i,j}]^T \in R^{p_i}$ $(j = 1,2,\ldots,n_\Sigma)$ are respectively the jth column of matrices F_i and L_i.

Incorporating equations (6.24) and (6.25) into equation (6.23b) and after some rearrangement gives the following matrix-vector equation:

$$\Phi l = f, \tag{6.26a}$$

where

$$\Phi = [0_{\{m_i(n_\Sigma - r_i)\}\times(p_i r_i)} \quad \Omega], \tag{6.26b}$$

$$\Omega = diag\,\{K_i\} \in R^{m_i(n_\Sigma - r_i)\times p_i(n_\Sigma - r_i)}, \tag{6.26c}$$

$$l = \begin{bmatrix} l_1^T & l_2^T & \cdots & l_{n_\Sigma}^T \end{bmatrix}^T \in R^{p_i n_\Sigma}, \tag{6.26d}$$

$$f = \begin{bmatrix} f_{r_i+1}^T & f_{r_i+2}^T & \cdots & f_{n_\Sigma}^T \end{bmatrix}^T \in R^{m_i(n_\Sigma - r_i)}, \tag{6.26e}$$

In equation (6.26b), $0_{\{m_i(n_\Sigma - r_i)\}\times(p_i r_i)}$ is a zero matrix of dimension $\{m_i(n_\Sigma - r_i)\}\times(p_i r_i)$.

Let us now consider equation (6.22b). With $E_i \in R^{p_i \times p_i}$ chosen to have a desired stable eigenstructure and by some simple rearrangement, (6.22b) can be put in a matrix-vector form as

$$\Psi\, l = 0, \tag{6.27a}$$

where matrix $\Psi \in R^{p_i(n_\Sigma - r_i)\times p_i n_\Sigma}$ is given by

$$\Psi = \begin{bmatrix} a_{1,r_i+1}I_{p_i} & a_{2,r_i+1}I_{p_i} & \cdots & (a_{r_i+1,r_i+1}I_{p_i} - E_i) & \cdots & a_{n_\Sigma-1,r_i+1}I_{p_i} & a_{n_\Sigma,r_i+1}I_{p_1} \\ a_{1,r_i+2}I_{p_i} & a_{2,r_i+2}I_{p_i} & \cdots & \cdot & \cdots & \cdot & a_{n_\Sigma,r_i+2}I_{p_i} \\ \cdot & \cdot & \cdots & \cdot & \cdots & \cdot & \cdot \\ a_{1,n_\Sigma-1}I_{p_i} & a_{2,n_\Sigma-1}I_{p_i} & \cdots & \cdot & \cdots & (a_{n_\Sigma-1,n_\Sigma-1}I_{p_1} - E_i) & a_{n_\Sigma,n_\Sigma-1}I_{p_i} \\ a_{1,n_\Sigma}I_{p_i} & a_{2,n_\Sigma}I_{p_i} & \cdots & a_{r_i+1,n_\Sigma}I_{p_i} & \cdots & a_{n_\Sigma-1,n_\Sigma}I_{p_i} & (a_{n_\Sigma,n_\Sigma}I_{p_i} - E_i) \end{bmatrix}, \tag{6.27b}$$

and $a_{j,k}$ denotes the (j,k)-element of matrix A. Similarly, (6.20) can be put in a matrix-vector form as

$$\Theta\, l = 0, \tag{6.28a}$$

where $$\Theta = \begin{bmatrix} b_{1,1}I_{p_i} & b_{2,1}I_{p_i} & \cdots & b_{n_\Sigma,1}I_{p_i} \\ b_{1,2}I_{p_i} & b_{2,2}I_{p_i} & \cdots & b_{n_\Sigma,2}I_{p_i} \\ \cdot & \cdot & \cdots & \cdot \\ b_{1,(m_\Sigma-m_i)}I_{p_i} & b_{2,(m_\Sigma-m_i)}I_{p_i} & \cdots & b_{n_\Sigma,(m_\Sigma-m_i)}I_{p_i} \end{bmatrix} \in R^{\{p_i(m_\Sigma-m_i)\}\times(p_i n_\Sigma)}, \tag{6.28b}$$

and $b_{j,k}$ denotes the (j,k)-element of matrix B_r.

The problem is thus concluded in that of finding a feasible solution to equation:

$$\begin{bmatrix} \Phi \\ \Psi \\ \Theta \end{bmatrix} l = \begin{bmatrix} f \\ 0 \\ 0 \end{bmatrix}. \tag{6.29}$$

Note that $\{m_i(n_\Sigma - r_i) + p_i(n_\Sigma - r_i) + p_i(m_\Sigma - m_i)\}$ linear simultaneous equations with $p_i n_\Sigma$ unknowns can be exactly solved if

$$\{m_i(n_\Sigma - r_i) + p_i(n_\Sigma - r_i) + p_i(m_\Sigma - m_i)\} \le p_i n_\Sigma, \tag{6.30}$$

the observer order p_i should therefore be chosen such that $p_i \ge \dfrac{m_i(n_\Sigma - r_i)}{r_i + m_i - m_\Sigma}$. If an exact solution to (6.29) is obtained then with E_i selected to be Hurwitz, error vectors $e_i(t)$ asymptotically approach zero. The local control laws (6.12a) will therefore reproduce asymptotically the global control (6.9).

Exact solutions to (6.12) may not always, however, be found, especially if low orders p_i of the observers (6.12b) are preferred. Alternatively, an approximate solution is procedure for solving matrices K_i and L_i [Ha and Trinh, 2004]. The procedure using the Moore-Penrose inverse and singular value decomposition of matrices involves the formulation and solution of an optimisation problem, which will minimise the norm of the error between the two sides of equation (6.26) and (6.27). It is shown that the error norm of these two equations will determine the overall closed-loop stability of the system. The advantages of the approach include

- the observers are completely decentralised in that each local control station uses locally available information only to generate the local control input signal, and hence, no information transfer among the local controllers required; and
- the order of the each local observer can be selected from a lowest value.

6.4 Design illustration and results

6.4.1 Modelling

For the illustration purpose let us consider a simple case of two mobile robots controlled in a 2-D formation parallel to the horizontal axis with a common abscissa and a given average ordinate in a global Cartesian coordinate system. Here, the formation can be described by $x_1 = x_2$, $\frac{y_1 + y_2}{2} = 0$, and $\theta_1 = \theta_2 = 0$, where (x_1, y_1, θ_1) and (x_2, y_2, θ_2) are respectively the position and orientation of robot 1 and robot 2.

The global state vector of the form (6) is chosen as $S = [\bar{\theta}_1\ \bar{\theta}_2\ h_1\ h_2]^T$, where $\bar{\theta}_1 = \theta_1, \bar{\theta}_2 = \theta_2$, and the platoon level functions are $h_1 = x_1 - x_2 + \alpha(\theta_1 - \theta_2)$ and $h_2 = y_1 + y_2 + \beta(\theta_1 + \theta_2)$. Here functions h_1 and h_2 contain respectively the global information of the horizontal distance error between the robots, and the average value of the formation vertical position, and α, β represent the level of perturbation in distance measurements due to the robot orientation

Linearising about the formation trajectory with $v = 2, \theta = 0$, one can obtain the system equation of the form (6.15):

$$\dot{S} = AS + B_1 u_1 + B_2 u_2, \tag{6.31}$$

where the local control inputs are

$$u_1 = \begin{bmatrix} v_1 - 2 \\ \omega_1 \end{bmatrix}, \quad u_2 = \begin{bmatrix} v_2 - 2 \\ \omega_2 \end{bmatrix},$$

(v_1, ω_1) and (v_2, ω_2) are the translational and angular velocities of robot 1 and robot 2 respectively, and

$$A=\begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\2&2&0&0\end{bmatrix},\quad B_1=\begin{bmatrix}0&1\\0&0\\1&\alpha\\0&\beta\end{bmatrix},\quad B_2=\begin{bmatrix}0&0\\0&1\\-1&-\alpha\\0&\beta\end{bmatrix}.$$

Here the decentralized information structure of the form (6.8) includes $y_1=[\theta_1\ h_1\ h_2]^T$ and $y_2=[\theta_2\ h_1\ h_2]^T$.

6.4.2 Observer design

The global controller can be designed by using any available techniques in the control theory [Nguyen *et al.*, 2004]. For example, with $\alpha=0.2,\beta=0.5$, placing the closed-loop eigenvalues at {-1.5-1.5i; -1.5+1.5i; -0.8; -0.5} for the feedback control $u=FS$ yields

$$F=\begin{bmatrix}0.0098&-0.1411&-0.25&-0.0568\\-1.1826&-0.1268&0&-0.7958\\-0.0098&0.1411&0.25&0.0568\\-1.0851&-1.5376&0&-1.3638\end{bmatrix}.$$

By applying the proposed method, observer–based decentralized observers of the form (6.12) can be obtained with the chosen matrix $E_i=\begin{bmatrix}-4&0\\0&-4\end{bmatrix}$:

Robot 1

$$K_1=\begin{bmatrix}0.07055&0.07055\\0.0634&0.0634\end{bmatrix},\quad L_1=\begin{bmatrix}0&2&0&-1\\0&2&0&-1\end{bmatrix},$$
$$W_1=\begin{bmatrix}-0.25&-0.339&0.0098\\0&-1.0494&-1.1826\end{bmatrix},\quad G_1=\begin{bmatrix}0&8&4\\0&8&4\end{bmatrix}.$$

Robot 2

$$K_2 = \begin{bmatrix} 0.0049 & 0.0049 \\ 0.054255 & 0.54255 \end{bmatrix}, \quad L_2 = \begin{bmatrix} 0 & 2 & 0 & -1 \\ 0 & 2 & 0 & -1 \end{bmatrix},$$
$$W_2 = \begin{bmatrix} 0.25 & 0.0372 & 0.1411 \\ 0 & -3.534 & -1.5376 \end{bmatrix}, \quad G_2 = \begin{bmatrix} 0 & 8 & 4 \\ 0 & 8 & 4 \end{bmatrix}$$

6.4.3 Simulation results

The objective of the simulation is to validate the proposed approach for two mobile robots based on the model presented in Section 6.4.1.

With the specified formation, the desired positions and orientations are $x_{1d} = x_{2d}, y_{1d} + y_{2d} = 0, \theta_{1d} = \theta_{2d} = 0$. The following initial condition is chosen in our simulation:

$$\text{Robot 1:} \begin{pmatrix} x_1(0) = -6(cm) \\ y_1(0) = 1(cm) \\ \theta_1(0) = 1(rad) \end{pmatrix}, \text{Robot 2:} \begin{pmatrix} x_2(0) = 15(cm) \\ y_2(0) = 9(cm) \\ \theta_2(0) = 2(rad) \end{pmatrix}$$

To compare, Figure 6.2 depicts the trajectories of robots with centralized and decentralized controllers. The global states $(\theta_1, \theta_2, h_1, h_2)$ of the systems are shown in Figure 6.3, when controlled in both centralized and decentralized manner. Figures 6.4 presents some snapshots over the time scale [0 15sec] of the multi-agent system with indices denoting the time points and dash lines representing the desired trajectories of the robots in the formation.

Simulation has been conducted for three robots moving in wedge or parallel line/column formations. Fig. 6.5 shows the trace of three robots in various formations, where changes are observed from a column to circular lines, circular to prismatic lines, and then back to the column formation.

It is clear from simulation results that the multi-robot system with the proposed decentralized controllers can form and maintain this simple desired formation. For controlling a more complicated formation, a piecewise linearization technique would be required. Experimental work has been conducted to verify the proposed technique, as reported in the next section.

6.4.4 Experimental results

Experimental platforms used for testing are the Amigobots. A photograph of the two mobile robots when maintaining a line formation is shown in Figure 6.6. Range-finding is handled by eight sonars mounted on the side of the robot, six in the fron and two in the back. Shaft encoders track its local position. By using differential drive and the nearly holonomic design, the robot mobility is acceptable over carpet edges and small sills. Information of sensing, motor and power monitoring and control is sent in packets over the wireless or tethered RS232 serial connection to PCs [Nguyen *et al.*, 2006a]. Specifications of the testbed- Amigo Mobile Robot- are given in Appendix A.

As described in the simulation, the objective of our experiments is to demonstrate the decentralized control algorithm for two Amigobots in entering and maintaining a simple formation, from an arbitrary initial condition. Here, the control actions together with platoon level functions were computed from local information transferred from robots to the PC. The control inputs (translational and angular velocities) of each robot were computed in a decentralized manner. These signals were sent back through the wireless network to each robot. The control algorithm was programmed in C++ using ARIA (Advanced Robotics Interface for Applications) class. More details of ARIA class are given in Appendix B.

Figures 6.7 and 6.8 show the time responses of position X, Y, orientation θ and trajectories, in the global coordinates of the two Amigobots forming a line from initial conditions:

$$x_1(0) = -400mm, \quad y_1(0) = -500mm, \quad \theta_1(0) = 90^0,$$
$$x_1(0) = -600mm, \quad y_1(0) = 600mm, \quad \theta_1(0) = 0^0,$$

respectively for Robots 1 and 2, where the formation was successfully established after 2.5 minutes with a line length of 0.8 metre.

In our experiments, the line formation for the two robots was two symmetrical straight trajectories parallel to the horizontal axis, as in the simulated results shown in Figures

6.2 - 6.4. The two robots should enter the formation parallel trajectories, located equidistantly to the horizontal axis.

A column formation can also be formed from an arbitrary location of Robots 1 and 2, say from:

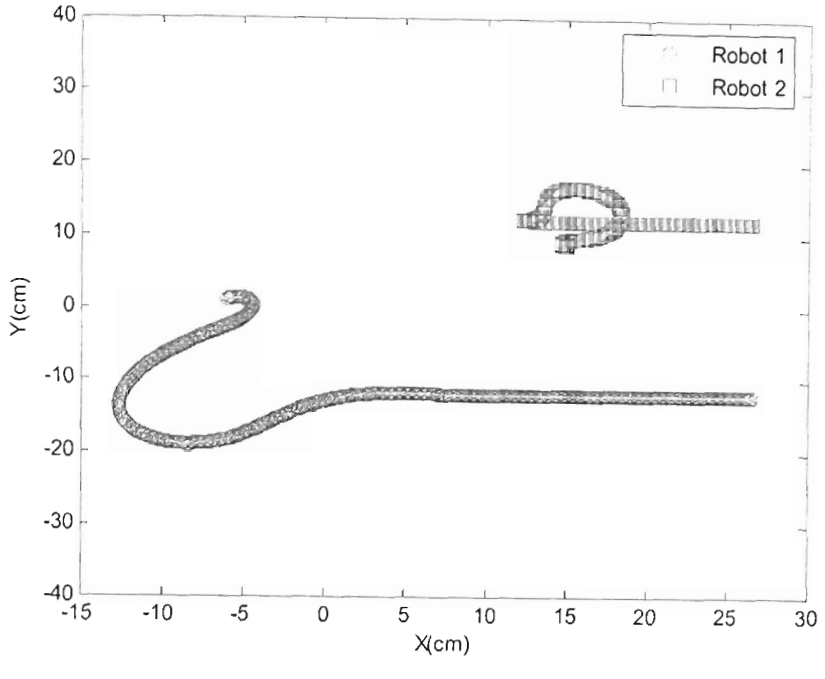
$$x_1(0) = -400mm, \quad y_1(0) = -500mm, \quad \theta_1(0) = 90^0,$$
$$x_1(0) = -600mm, \quad y_1(0) = 600mm, \quad \theta_1(0) = 0^0,$$

as shown in Figures 6.9 and 6.10, where the column width of 0.9 metre was maintained after 160 seconds.

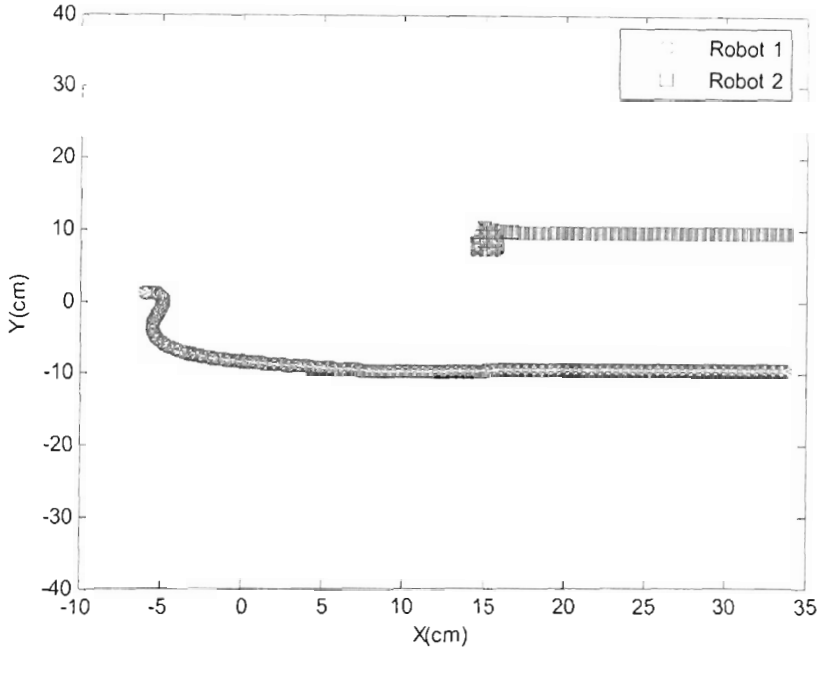
Experimental results show that these simple robotic formations are formed and maintained successfully with two Amigobots. However, a small trajectory tracking errors may occur due to errors in position information, transferred from the encoders to the PC through the wireless communication. Tracking accuracy can be improved by using better sensors and by incorporating further navigation assistive algorithms for robot localization.

6.5 Conclusion

This chapter has presented a solution to the problem of controlling a platoon of robots in a formation under a complete decentralized information structure. The proposed approach exploits decentralised linear functional observers to implement a suitable global feedback control law. Each local controller takes some global information of the formation from an exogenous unit and only local output measurements. The design technique is illustrated through the control of groups of mobile robots with simulation results provided. Experimental results reported illustrate the validity of the proposed technique for two Amigo robots in entering and maintaining simple formations from an arbitrary initial position.



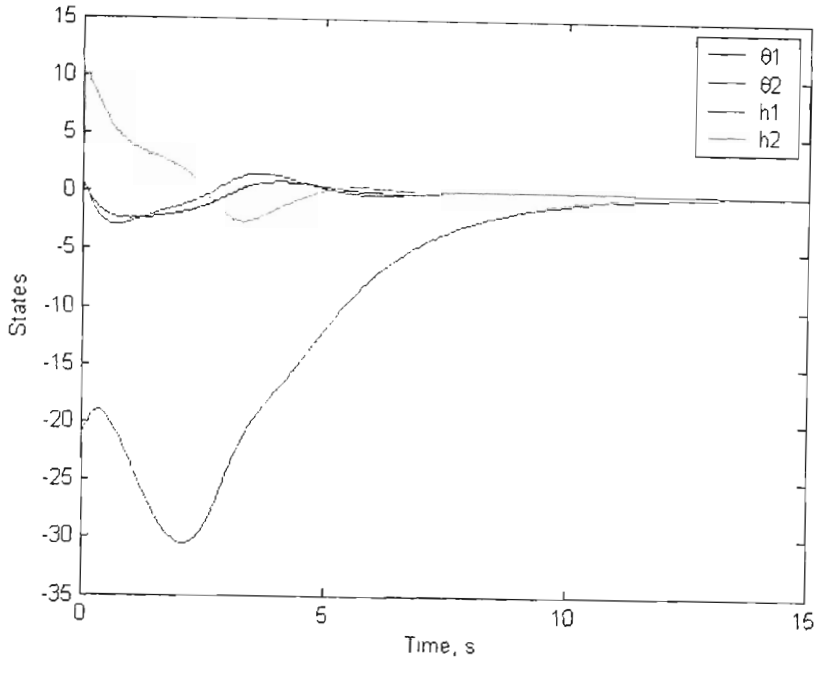
a)



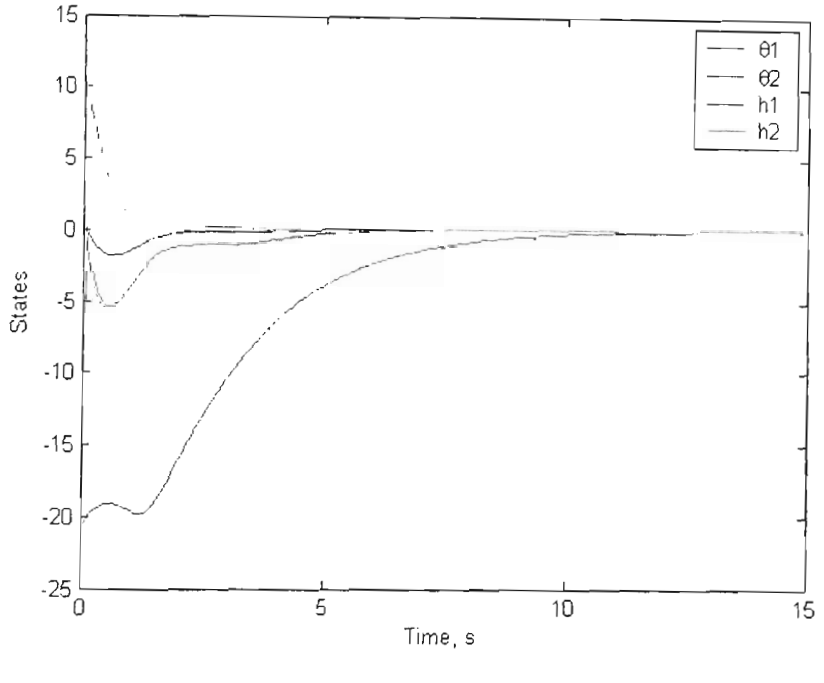
b)

Figure 6.2: Simulation 6.1 results - Robot trajectories

a) with centralised control b) with decentralised control



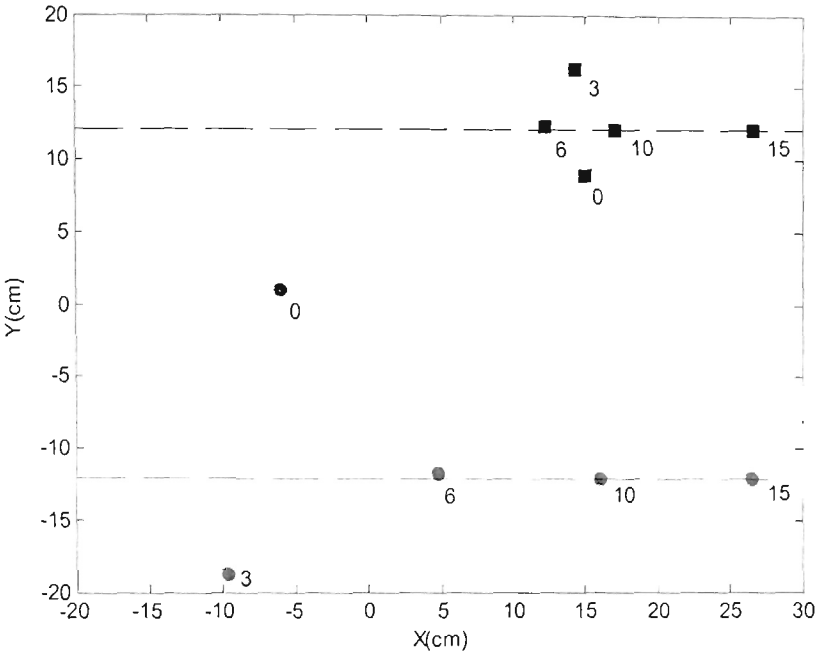
a)



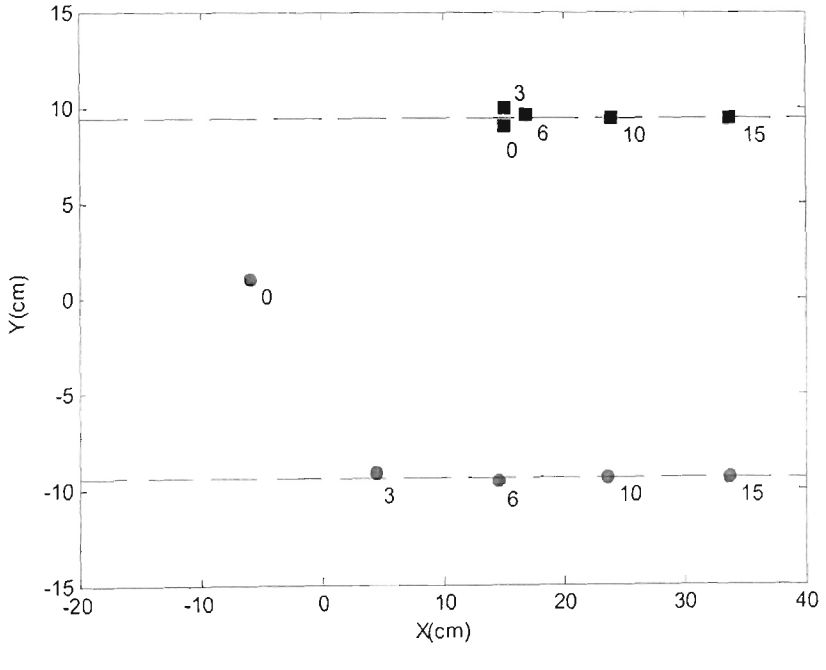
b)

Figure 6.3: Simulation 6.1 results - Global states of the system

a) with centralised control b) with decentralised control



a)



b)

Figure 6.4: Simulation 6.1 results - Snapshots over time

a) with centralised control b) with decentralised control

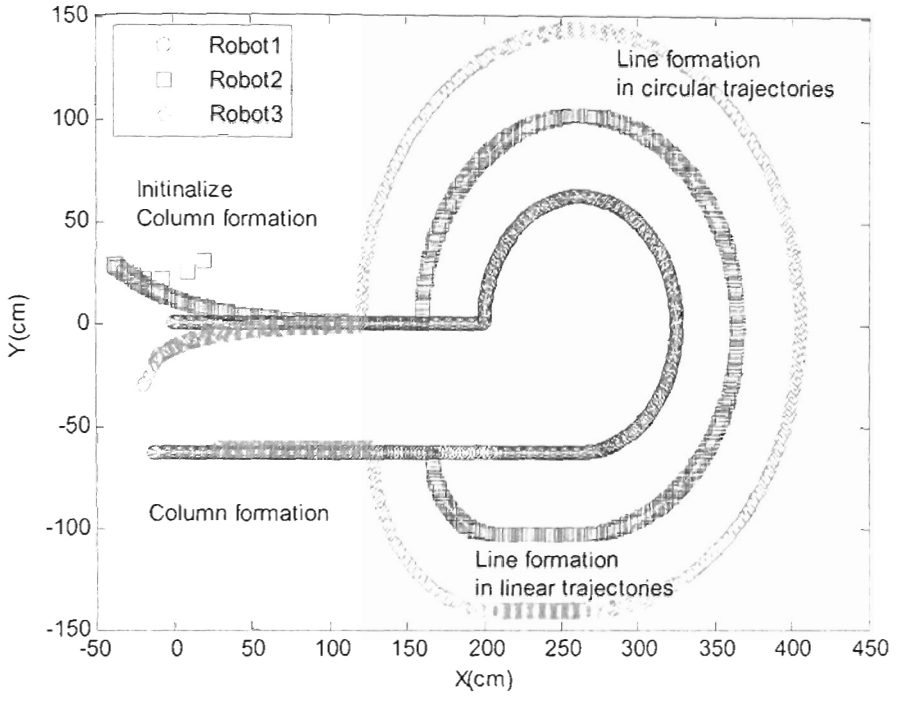


Figure 6.5: Simulation 6.2 results - Three robots in various formations

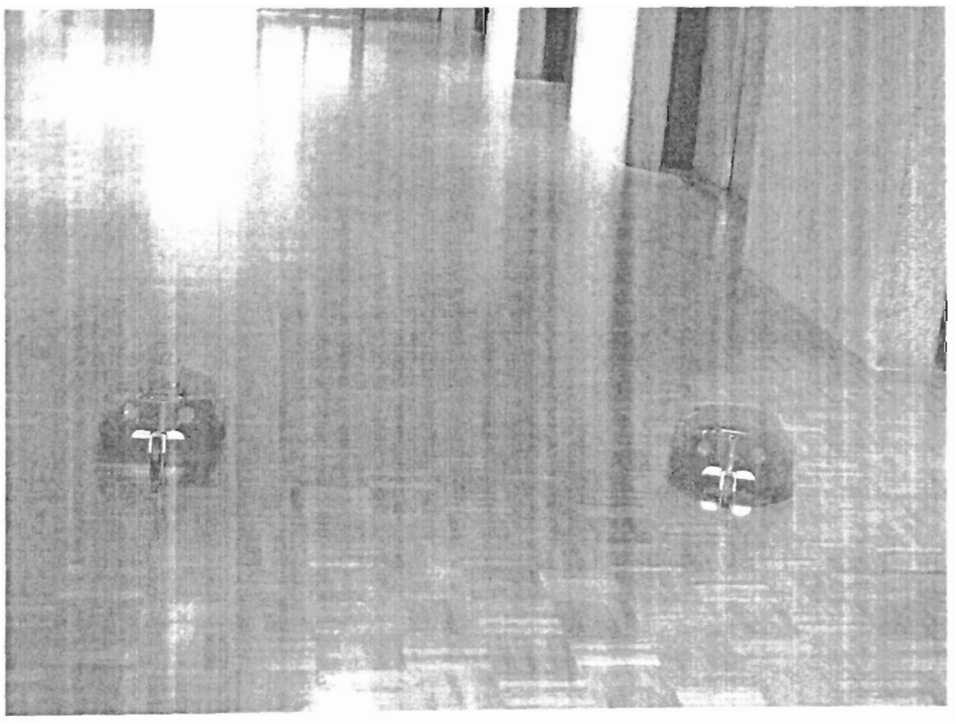
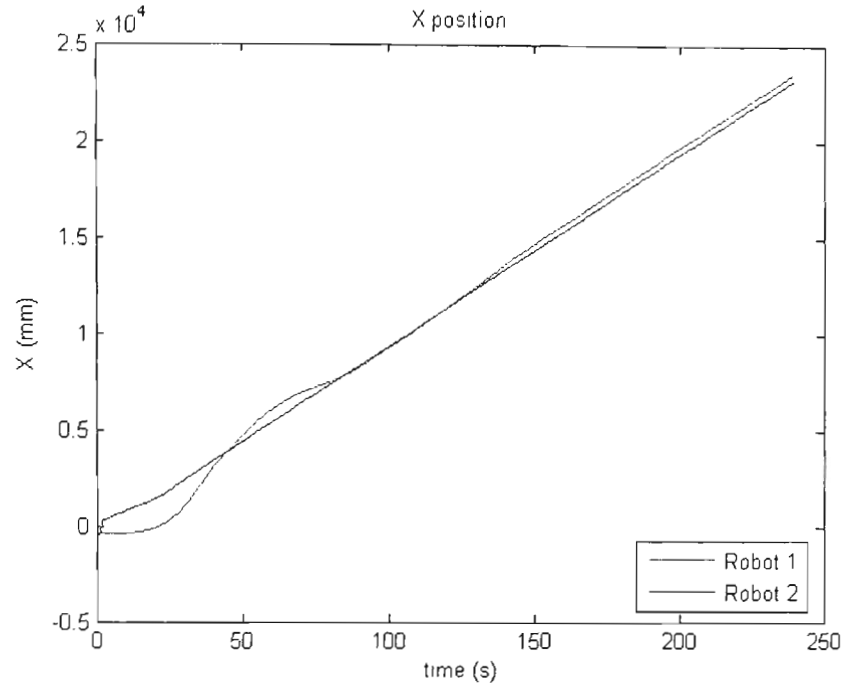
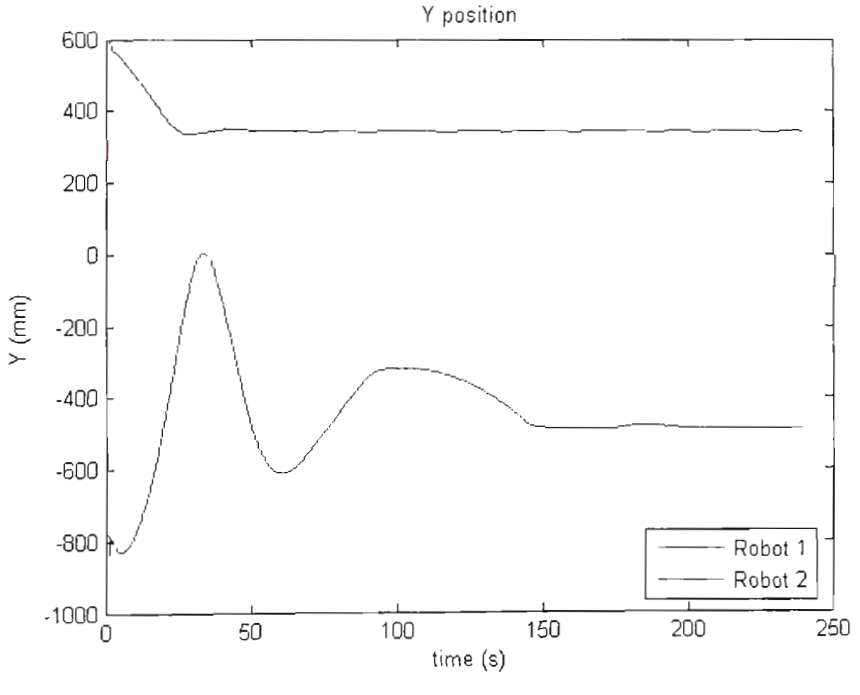


Figure 6.6: Two Amigobots performing a line formation.



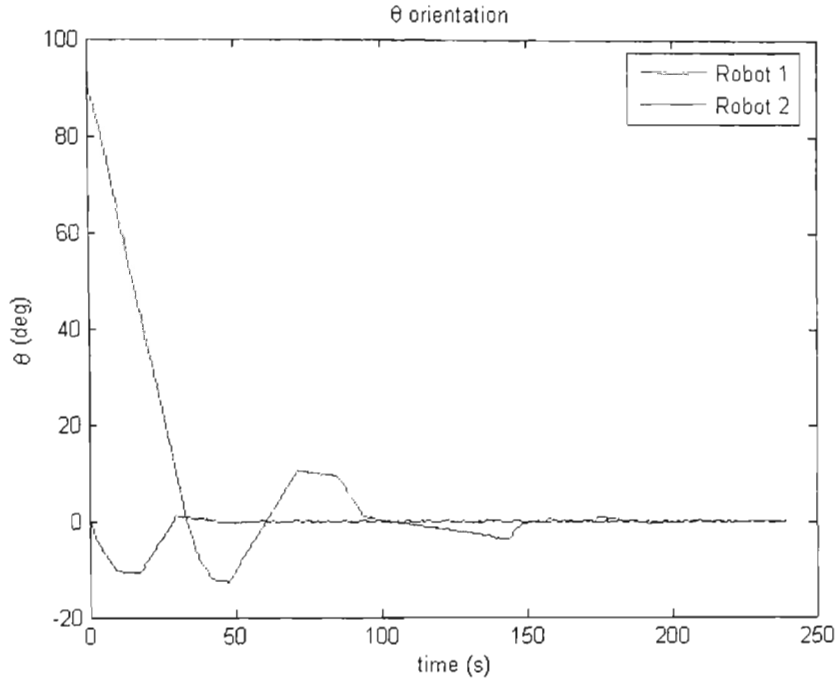
a)



b)

Figure 6.7: Experimental results - Line formation: X, Y position

a) X position b) Y position



a)

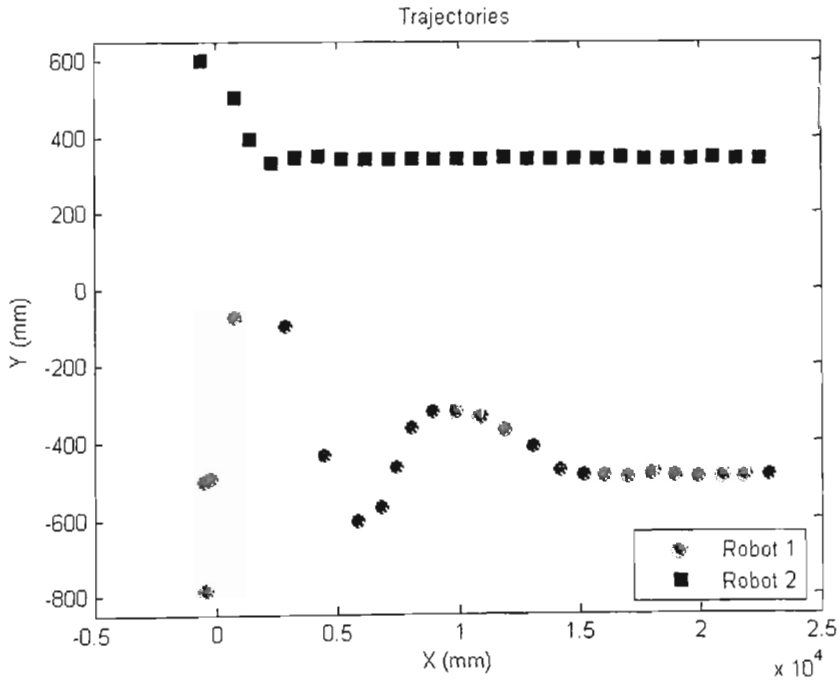
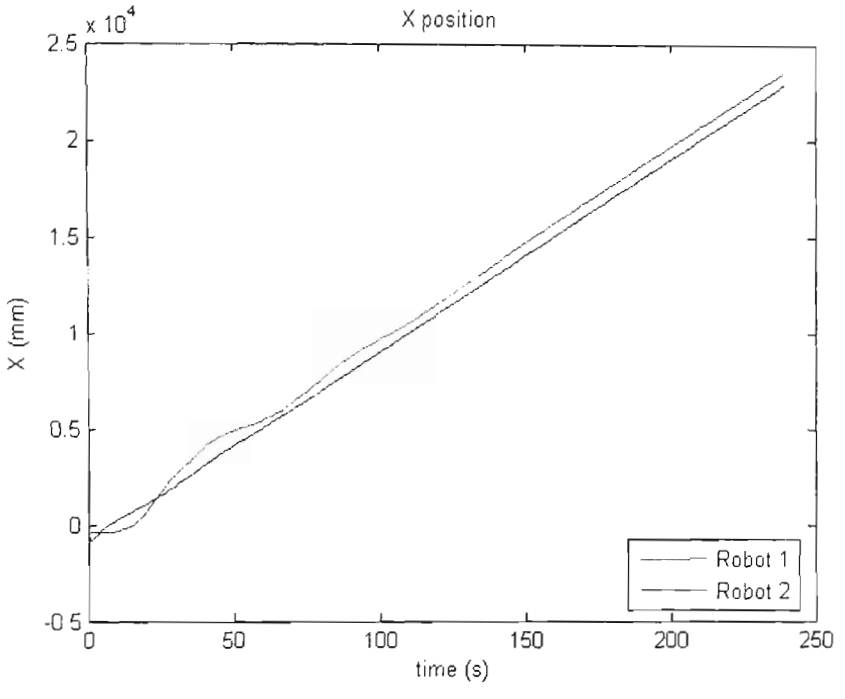
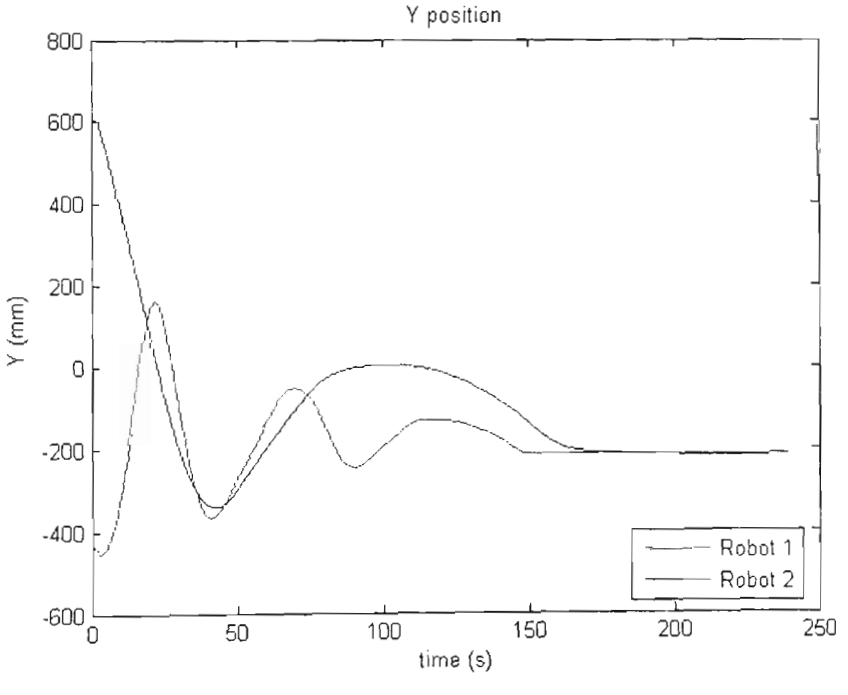


Figure 6.8: Experimental results - Line formation: θ orientation and trajectories

a) θ- orientation b) Trajectories



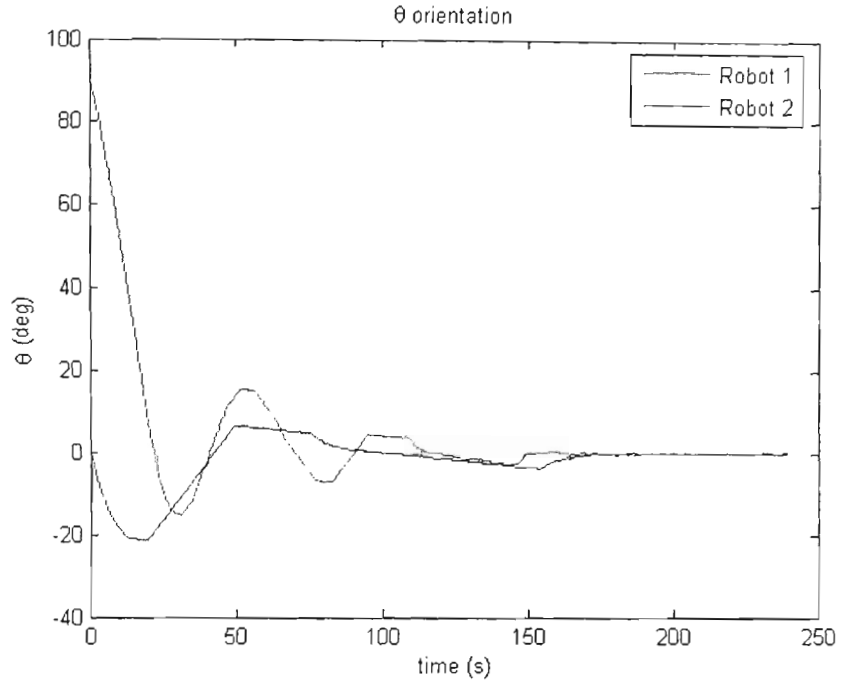
a)



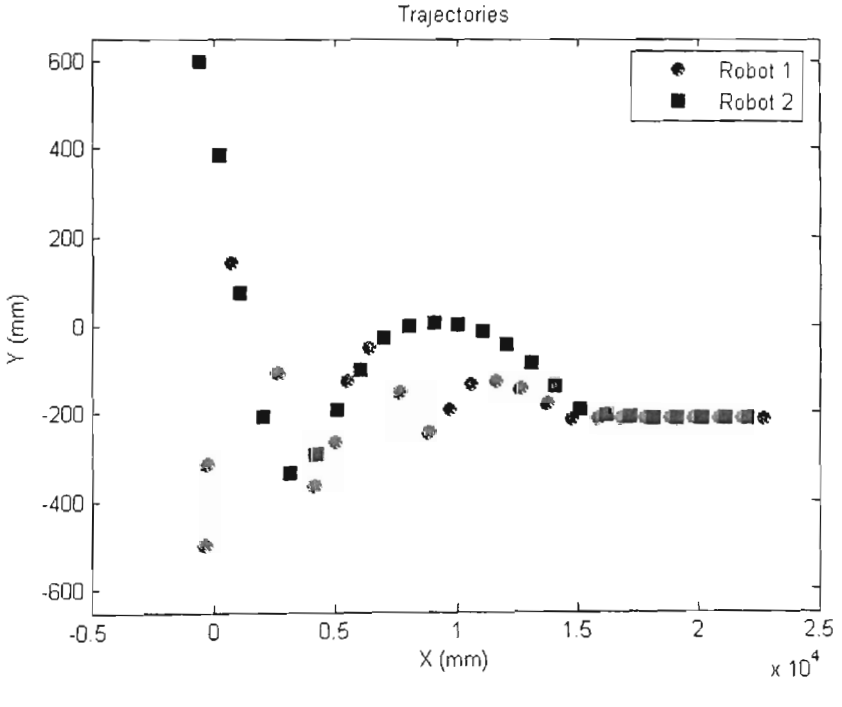
b)

Figure 6.9: Experimental results - Column formation: X, Y position

a) X position b) Y position



a)



b)

Figure 6.10: Experimental results - Column formation: θ orientation and trajectories: a) θ- orientation b) Trajectories

Chapter 7

Thesis summary and conclusion

7.1 Introduction

This chapter summarises the work presented throughout the thesis and draws its conclusions. In Section 7.2 the contents of each chapter are summarised, whilst the contributions of the thesis are reviewed in Section 7.3. Some future research issues are suggested in Section 7.4. A conclusion of the thesis is given in Section 7.5

7.2 Chapter summary

Chapter 1 presents the aim of this thesis: the development of decentralised controllers to initialise and maintain a desired formation of multiple mobile robots in an obstacle-free environment. An overview of the work covered in this project is also included.

Together with the review of the commonly-used models of mobile robots, a short survey of control approaches available for analysis and design of robotic formation is given in Chapter 2.

The design of controllers to establish and maintain a desired leader-follower configuration between two mobile robots is concerned in Chapter 3. After reviewing some existing control laws: $l-\psi$, *separation-bearing*, and *virtual robot tracking*

control, a new *virtual head robot tracking* control is proposed. This proposed controller can overcome limitations of *virtual robot tracking* control in forming a line formation and in completely collision avoiding between robots. A standard I/O linearisation technique is used to generate a control law for follower robot j such that its head robot can track the virtual robot of leader i with the position errors decreasing monotonically. This controller can be used as a basic controller in leader-follower strategies to solve the robotic formation problems. Circumstances of possibility of collisions between robots concerning the "critical area" are also analysed. Collisions in those cases can be avoided by using a reactive control scheme with 3PLL control, which will be developed in Chapter 4. Simulation results show some cases the new proposed controller can remedy shortcomings of *virtual robot tracking* controller to obtain a desired configuration between two mobile robots.

Chapter 4 deals with the problem of collision avoidance in a group of three mobile robots. Some control laws for formation control, which have a capability of collision avoidance, such as original *l-l*, *separation-separation*, and modified *l-l* control are described with their singularity. *Three point l-l* control is proposed to alleviate the singularity. The idea is when the singularity happens, an appropriate controller shall be chosen among three *l-l* controllers related to three virtual points around the centre of the concerning robot. Finally, an algorithm combining *virtual head robot tracking* and *three point l-l* control incorporated with a reactive control scheme is proposed for formation control of a group of three mobile robots without inter-robot collisions. Simulations with a group of three mobile robots are used to demonstrate the validity of the proposed approach.

Formation initialisation for a group of N mobile robots is concerned in Chapter 5. Firstly, two reactive control schemes to deal with potential collisions between two and three mobile robots are presented using *three point l-l* control. Based on the reactively controlled VHRT-3PLL, a step-by-step procedure is proposed to initialise a desired formation for a group of N mobile robots. Robots in the group sequentially participate to the process of formation initialisation so that collision avoidance among them can be guaranteed. Weighted, directed graphs are used to model formations established through the step-by-step procedure. A method to enumerate all allowable control

graphs for given indexed N mobile robots is also presented. Each control graph could be recalled based on the constraints in the environment, positions of robots in the group and the motion of the lead robot. Extensive simulations with five robots validate the capability of forming formations for a group of more than three robots without collisions among them.

To accommodate the restriction in information exchange, a new observer-based decentralised control approach to robotic formation is proposed in Chapter 6. The proposed solution is based on the design of functional observers to estimate asymptotically the global state-feedback control signals by using the corresponding local output information and some exogenous global functions. The proposed technique is tested through simulation and experiments for the control of groups of Amigo mobile robots.

7.3 Thesis contribution

7.3.1 Virtual head robot tracking control

The first contribution of this thesis is the development of *virtual head robot tracking* control to establish a desired configuration between two mobile robots. Motivated by existing tracking control laws in leader-follower strategies for robotic formation, the proposed controllers can be applied to obtain leader-follower configurations as foundations of desired formations. Collisions between robots could be avoided in most of circumstances, except cases related to "critical area" which can be dealt by using an appropriate reactive control scheme with *three point l-l* control.

7.3.2 Three point *l-l* control and an algorithm for formation control of a group of three mobile robots

The next contribution of this thesis is the development of *three point l-l* control to avoid collisions among robots. In fact, existing *l-l* -based control laws all subject to a singularity when three concerning robots lie on the same line connecting them. *Three point l-l* control can deal with this singularity by switching among three designed *l-l*

controllers. Based on *virtual head robot tracking* and *three point l-l* control, an algorithm for formation control of a group of three mobile robots is proposed with the capability of establishment any desired formation without inter-robot collisions. This approach can be thought as an effective solution to formation problem for a group of three mobile robots in an obstacle-free environment.

7.3.3 Formation initialisation for a group of N mobile robots

A step-by step procedure for the process of formation initialisation has been proposed. Classifying robots in a group into *active* and *inactive* sub-group, this procedure under some assumptions can lessen possibilities of collisions among robots. Two proposed reactive control schemes are useful when the possibility of collision among robots can be detected. In addition, weighted digraphs have been used to model group formations which are established by following the proposed procedure. A library of allowable control graphs can be created and used for choosing an appropriate graph to deal with the change in the environment or in motion of the lead robot.

7.3.4 Observer-based decentralised control approach

Finally, to accommodate the restriction in information exchange, a decentralized approach is proposed to implement the global feedback controller for the robot moving in the formation by using linear functional observers. In this decentralised control model, each local controller takes some global information of the formation from an exogenous unit and only local output measurements. This approach gives a group of mobile robots in formation the scalability, i.e. the flexibility to add robots into a formation to a large number of members, in the condition of communications bandwidth limitations. Both simulation and experimental results conducted on Amigo mobile robots confirm the validity of the proposed technique.

7.4 Future work

From the research reported in this thesis, the following aspects are suggested for further investigation:

- *Robotic formation control with learning capabilities*: It is proposed in the thesis that *virtual head robot tracking* and *three point l-l* control laws can be suggested to establish and maintain a desired formation of multiple mobile robots without inter-robot collisions. However, when initial errors between real and desired positions of robots are quite large, the calculated velocities may exceed the maximum admissible velocities of real mobile robots. To implement the proposed control laws into practice, it is necessary to consider the dynamical interactions of real robots with environment by introducing schemes for control structures and tuning control parameters. The selection and learning of tuning parameters is open issue for further investigation.

- *Assigning desired positions in formations*: Under some assumptions, a step-by-step procedure can be followed to initialise formations for N indexed given mobile robots. An algorithm to assign desired positions in formations for robots in a group is worthy of further investigation to lessen possibilities of collisions among robots in the process of formation initialisation in general practical scenarios. Better knowledge of the environment and global coordinates of robots would significantly help for such algorithm's performance.

- *Obstacle avoidance*: Following the proposed step-by-step procedure for formation initialisation, at each step *inactive* robots, which do not participate to the process of forming desired geometrical shape, can play the roles of static obstacles. Therefore the main problem on obstacle avoidance in the framework of multi-robot coordination should be examined. Behaviour-based and potential filed techniques would be suitable in the present work for this purpose.

- *Decentralised control with nonlinear models of mobile robots*: The proposed observer-based decentralised control in Chapter 6 used linearised models of mobile robots around specific trajectories. The main disadvantages are the difficulties to determine the initial condition range of the robots from that the desired formation can be obtained. This problem may be overcome with the following considerations:

- To construct a nonlinear model of a group of mobile robots in a formation.

- To design nonlinear centralized controller for this group

- To design nonlinear-functional observers to estimate control inputs for each robot

A thorough examination of nonlinear control for multi agent systems is needed for that research.

7.5 Conclusion

This thesis presents original contributions towards a solution to the problem of formation initialisation and maintenance for multiple mobile robots in an obstacle-free environment with a careful consideration paid to the problem inter-robot collision avoidance. Once initialised, the formation can be controlled using an observer-based decentralised technique to satisfy communication bandwidth limitations. The theoretical work of the thesis is evaluated by extensive simulation of multiple mobile robots based on their kinematic models. The results obtained are also experimentally tested, in part, on a group of two Amigo mobile robots.

Formation control in an environment with possible incidents of obstacle collision will be an interesting research topic for future work.

Bibliography

[Anderson and Robbins, 1998] Anderson, M. R., and Robbins, A. C. "Formation flight as a cooperative game" *Proc. AIAA Guidance, Navigation, and Control Conference and Exhibit*, Boston, MA, USA, pp. 244-251, 10-12 August, 1998.

[Asada and Slotine, 1986] Asada, H., and Slotine, J.-J. E., 1986 *Robot Analysis and Control*, John Wiley & Sons, New York, USA.

[Balch and Arkin, 1995] Balch, T., and Arkin, R. C. "Motor Schema-based Formation Control for Multiagent Robot Teams" *Proc. International Conference on Multiagent Systems*, San Francisco, CA, USA, pp. 10-16, 12-14 June, 1995.

[Balch and Arkin, 1998] Balch, T., and Arkin, R., 1998 "Behavior-based formation control for multirobot teams" *IEEE Transactions on Robotics and Automation*, 14(6), pp. 926-939.

[Balch and Hybinette, 2000a] Balch, T., and Hybinette, M. "Behavior-based coordination of large-scale robot formations" *Proc. Fourth International Conference on MultiAgent Systems*, Boston, MA, USA, pp. 363-364, 10-12 July, 2000.

[Balch and Hybinette, 2000b] Balch, T., and Hybinette, M. "Social Potentials for Scalable Multi-Robot Formations" *Proc. IEEE International Conference on Robotics & Automation*, San Francisco, CA, USA, pp. 73-80, 24-28 April, 2000.

[Barhen *et al.*, 1989] Barhen, J., Gulati, S., and Zak, M., 1989 "Neutral Learning of Constrained Nonlinear Transformations" *IEEE Computer*, 22(6), pp. 67-76.

[Beard, 1998] Beard, R. W., 1998 "Architecture and algorithms for constellation control" *Technical Report*, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA.

[Beard and Hadaegh, 1998] Beard, R. W., and Hadaegh, F. Y. "Constellation templates: An approach to autonomous formation flying" *Proc. World Automation Congress*, Anchorage, Alaska, USA, pp. 177.1-177.6, May, 1998.

[Beard *et al.*, 2001] Beard, R. W., Lawton, J., and Hadaegh, F. Y., 2001 "A coordination architecture for spacecraft formation control" *IEEE Transaction on Control Systems Technology*, 9(6), pp. 777-790.

[Beard *et al.*, 2002] Beard, R. W., McLain, T. W., Goodrich, M. A., and Anderson, E. P., 2002 "Coordinated target assignment and intercept for unmanned air vehicles" *IEEE Transactions on Robotics and Automation*, 18(6), pp. 911-922.

[Bekey, 2005] Bekey, G. A., 2005 *Autonomous Robots: From Biological Inspiration to Implementation and Control*, The MIT Press, Cambridge, Massachusetts, London, England.

[Belta and Kumar, 2002] Belta, C., and Kumar, V. "Trajectory design for formations of robots by kinematic energy shaping" *Proc. IEEE International Conference on Robotics & Automation*, Washington DC, USA, pp. 2593-2598, 11-15 May, 2002.

[Camazine *et al.*, 2001] Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraulaz, G., and Bonabeau, E., 2001 *Self-Organisation in Biological Systems*, Princeton University Press, NJ, USA.

[Canudas de Wit *et al.*, 1996] Canudas de Wit, C., Siciliano, B., and Bastin, G., 1996 *Theory of Robot Control*, Springer-Verlag, New York, USA.

[Cao *et al.*, 2002] Cao, Z., Tan, M., Wang, S., Fan, Y., and Zhang, B. "The optimization research of formation control for multiple mobile robots" *Proc. The Fourth World Congress on Intelligent Control and Automation*, Shanghai, China, pp. 1270-1274, 10-14 June, 2002.

[Cao *et al.*, 2003] Cao, Z., Xie, L., Zhang, B., Wang, S., and Tan, M. "Formation constrained multi-robot system in unknown environments" *Proc. IEEE*

International Conference on Robotics and Automation, Taipei, Taiwan, pp. 735-740, 10-14 September, 2003.

[Carpin and Parker, 2002] Carpin, S., and Parker, L. E. "Cooperative Leader Following in a Distributed Multi-Robot System." *Proc. IEEE International Conference on Robotics & Automation*, Washington, DC, USA, pp. 2994-3001, 11-15 May, 2002.

[Chen and Luh, 1994] Chen, Q., and Luh, J. Y. S. "Coordination and Control of a Group of Small Mobile Robots" *Proc. IEEE International Conference on Robotics & Automation*, San Diego, CA, USA, pp. 2315-2320, 8-13 May, 1994.

[Chen and Wang, 2005] Chen, Y. Q., and Wang, Z. "Formation Control: A review and New Consideration" *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Alberta, Canada, pp. 3664-3669, 2-6 August, 2005.

[Cowan *et al.*, 2003] Cowan, N., Shakerina, O., Vidal, R., and Sastry, S. "Vision-based follow-the-leader" *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, USA, pp. 1796-1801, 27-31 October, 2003.

[Das *et al.*, 2002] Das, A. K., Fierro, R., Kumar, V., Ostrowski, J. P., J.Spletzer, and Taylor, A. J., 2002 "A vision - Based Formation Control Framework" *IEEE Transactions on Robotics and Automation*, 18(5), pp. 813-825.

[De Queiroz *et al.*, 2000] De Queiroz, M. S., Kapila, V., and Yan, Q., 2000 "Adaptive nonlinear control of multiple spacecraft formation flying" *Journal of Guidance, Control, and Dynamics*, 23(3), pp. 385-390.

[Desai, 2001] Desai, J. P. "Modelling Multiple Teams of Mobile Robots: A Graph Theoretic Approach" *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, Hawaii, USA, 29 October-03 November, 2003.

[Desai, 2002] Desai, J. P., 2002 "A Graph Theoretic Approach for Modelling Mobile Robot Team Formations" *Journal of Robotic Systems*, 19(11), pp. 511-525.

[Desai *et al.*, 1998] Desai, J. P., Ostrowski, J. P., and Kumar, V. "Controlling formations of multiple mobile robots" *Proc. IEEE International Conference on Robotics & Automation*, Leuven, Belgium, pp. 2864 - 2869, 16-20 May, 1998.

[Desai *et al.*, 2001] Desai, J. P., Ostrowski, J. P., and Kumar, V., 2001 "Modelling and control of formations of non-holonomic mobile robots" *IEEE Transactions on Robotics and Automation*, 17(6), pp. 905-908.

[Dixon *et al.*, 2001] Dixon, W. E., Dawson, D. M., Zergeroglu, E., and Behal, A., 2001 *Nonlinear Control of Wheeled Mobile Robots*, Springer, London, England.

[Donald *et al.*, 2000] Donald, B., Gariepy, L., and Rus, D. "Distributed manipulation of multiple objects using ropes" *Proc. IEEE International Conference on Robotics and Automation*, San Francisco, CA, USA, pp. 450-457, 24-28 April, 2000.

[Dudek and Jenkin, 2000] Dudek, G., and Jenkin, M., 2000 *Computational Principles of Mobile Robotics*, Cambridge University Press, New York, USA.

[Dudenhoeffer and Jones, 2000] Dudenhoeffer, D. D., and Jones, M. P. "A formation behavior for large-scale micro-robot force deployment" *Proc. 2000 Winter Simulation Conference*, Orlando, FL, USA, pp. 972-982, 10-13 December, 2000.

[Egerstedt and Hu, 2001] Egerstedt, M., and Hu, X., 2001 "Formation constrained multi-agent control" *IEEE Transactions on Robotics and Automation*, 17(6), pp. 947 - 951.

[Erkin *et al.*, 2003] Erkin, B., Onur, S., and Erol, S., 2003 "A Review: Pattern Formation and Adaptation in Multi-Robot Systems" *Technical Report CMU-RI-TR-03-43*, Robotics Institute - Carnegie Mellon University, Pittsburgh, PA, USA.

[Feddema and Schoenwald, 2002] Feddema, J., and Schoenwald, D., 2002 "Decentralized control of cooperative robotic vehicles: Theory and Application" *IEEE Transactions on Robotics and Automation*, 18(5), pp. 852-864.

[Fierro *et al.*, 2001] Fierro, R., Das, A. K., Kumar, V., and Ostrowski, J. P. "Hybrid control of formations of robots" *Proc. IEEE International Conference on Robotics and Automation*, Seoul, Korea, pp. 157-162, 21-26 May, 2001.

[Fierro *et al.*, 2002] Fierro, R., Song, P., Das, A. K., and Kumar, V., 2002 "Cooperative Control of Robot Formations" Cooperative Control and Optimization, R. Murphey and P. M. Pardalos, eds., Kluwer Academic Publisher, pp. 73-93.

[Fox *et al.*, 2000] Fox, D., Burgard, W., Kuruppa, H., and Thrun, S., 2000 "A probabilistic approach to collaborative multi-robot localization" *Autonomous Robot*, 8(3), pp. 325-344.

[Fredslund and Mataric, 2002] Fredslund, J., and Mataric, M. J., 2002 "A general algorithm for robot formation using local sensing and minimal communication" *IEEE Transaction on Robotics and Automation*, 18(5), pp. 837-846.

[Gazi, 2003] Gazi, V. "Formation control of a multi-agent system using decentralized nonlinear servomechanism" *Proc. IEEE Conference on Decision and Control*, Maui, Hawaii, USA, pp. 2531-2536, 9-12 December, 2003.

[Ge *et al.*, 2004] Ge, S. S., H., F. C., and W., L. K. "Multi-robot formations: queues and artificial potential trenches" *Proc. IEEE International Conference on Robotics and Automation*, New Orleans, CA, USA, pp. 3345-3350, 26 April - 1 May, 2004.

[Gustavi and Hu, 2005] Gustavi, T., and Hu, X. "Formation Control for Mobile Robots with Limited Sensor Information" *Proc. IEEE International Conference on Robotics & Automation*, Barcelona, Spain, pp. 1791-1796, 18-22 April, 2005.

[Ha and Trinh, 2004] Ha, Q. P., and Trinh, H., 2004 "Observer-based Control of multi-agent systems under decentralised information structure" *International Journal of Systems Science*, 35(12), pp. 719-728.

[Ha *et al.*, 2005] Ha, H. M., Nguyen, A. D., and Ha, Q. P. "Controlling formation of multiple mobile robots with inter-robot collision avoidance" *Proc. Australasian*

Conference on Robotics and Automation, Sydney, Australia, pp. 1-8, 5-7 December, 2005.

[Hadaegh *et al.*, 1998] Hadaegh, F. Y., Lu, W. M., and Wang, P. K. C. "Adaptive control of formation flying spacecraft for interferometry" *Proc. IFAC Conference on Large Scale Systems*, Rio Patras, Greece, pp. 97-102, 1998.

[Hong *et al.*, 2001] Hong, S.-W., Shin, S.-W., and Ahn, D.-S. "Formation control based on artificial intelligence for multi-agent coordination" *Proc. IEEE International Symposium on Industrial Electronics*, Pulsan, Korea, pp. 429-434, 12-16 June, 2001.

[Hsu and Liu, 2005] Hsu, H. C., and Liu, A. "Applying a Taxonomy of Formation Control in Developing a Robotic System" *Proc. IEEE International Conference on Tools with Artificial Intelligence*, Hong Kong, China, pp. 3-10, 14-16 November, 2005.

[Jennings *et al.*, 1997] Jennings, J. S., Whelan, G., and Evans, W. F. "Cooperative search and rescue with a team of mobile robots" *Proc. IEEE International Conference of Advanced Robotics*, Monterey, CA, USA, pp. 192-200, 7-9 July, 1997.

[Jongusuk and Mita, 2001] Jongusuk, J., and Mita, T. "Tracking Control of Multiple Mobile Robots: A Case Study of Inter-Robot Collision-Free Problem" *Proc. IEEE International Conference on Robotics & Automation*, Seoul, Korea, pp. 2885-2890, 21-26 May, 2001.

[Kang *et al.*, 2000] Kang, W., Xi, N., and Sparks, A. "Formation control of autonomous agents in 3D workspace" *Proc. IEEE International Conference on Robotics and Automation*, San Francisco, CA, USA, pp. 1755-1760, 24-28 April, 2000.

[Kobayashi *et al.*, 2003] Kobayashi, F., Tomita, N., and Kojima, F. "Re-formation of Mobile Robots using Genetic Algorithm and Reinforcement Learning" *Proc. SICE Annual Conference*, Fukui University, Japan, pp. 2902-2907, 4-6 August, 2003.

[Koo and Shahruz, 2001] Koo, T. J., and Shahruz, S. M. "Formation of a Group of Unmanned Aerial Vehicles (UAVs)" *Proc. American Control Conference*, Arlington, VA, USA, pp. 69-74, 25-27 June, 2001.

[Kowalczyk, 2002] Kowalczyk, W. "Target assignment strategy for scattered robots building formation" *Proc. The Third International Workshop on Robot Motion and Control*, Bukowy Dworek, Poland, pp. 181-185, 9-11 November, 2002.

[Lawton *et al.*, 2003] Lawton, J. R. T., Beard, R. W., and Young, B. J., 2003 "A decentralised Approach to Formation Manoeuvers" *IEEE Transactions on Robotics and Automation*, 19(6), pp. 933-941.

[Lee and Li, 2003] Lee, D., and Li, P. Y. "Formation and Maneuver Control of Multiple Spacecraft" *Proc. American Control Conference*, Denver, Colorado, USA, pp. 278-283, 4-6 June, 2003.

[Lee *et al.*, 2005] Lee, C., Kim, M., and Kazadi, S. "Robot Clustering" *Proc. IEEE International Conference on Systems, Man and Cybernetics*, Big Island, Hawaii, USA, pp. 1449 - 1454, 10-12 October, 2005

[Lemay *et al.*, 2004] Lemay, M., Michaud, F., Letourneau, D., and Valin, J. M. "Autonomous Initialization of Robot Formations" *Proc. IEEE International Conference on Robotics & Automation*, New Orleans, LA, USA, pp. 3018-3023, 26 April - 1 May, 2004.

[Lewis and Tan, 1997] Lewis, M. A., and Tan, K. H., 1997 "High Precision Formation Control of Mobile Robots Using Virtual Structures" *Autonomous Robot*, 4(4), pp. 387-403.

[McInnes, 1995] McInnes, C. R., 1995 "Autonomous ring formation for a planar constellation of satellites" *Journal of Guidance, Control, and Dynamics*, 18(5), pp. 1215-1217.

[Michaud *et al.*, 2002] Michaud, F., Letourneau, D., Guilbert, M., and Valin, J. M. "Dynamic Robot Formations Using Directional Visual Perception" *Proc.*

IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, pp. 2740-2745, 30 September - 5 October, 2002.

[Monteiro and Bicho, 2002] Monteiro, S., and Bicho, E. "A dynamical systems approach to behavior-based formation control" *Proc. IEEE International Conference on Robotics and Automation*, Washington, DC, USA, pp. 2606-2611, 11-15 May, 2002.

[Nguyen *et al.*, 2004] Nguyen, A. D., Ha, Q. P., Huang, S., and Trinh, H. "Observer-Based Decentralized Approach to Robotic Formation Control" *Proc. Australasian Conference on Robotics and Automation (ACRA 2004)*, Canberra, Australia, pp. 1-8, 6-8 December, 2004.

[Nguyen *et al.*, 2006a] Nguyen, A. D., Ngo, V. T., Ha, Q. P., and Dissanayake, G., 2006 "Robotic Formation: Initialisation, Trajectory Planning, and Decentralised Control" *International Journal of Automation and Control*, accepted February, 2006

[Nguyen *et al.*, 2006b] Nguyen, A. D., Ha, Q. P., and Nguyen, H. T. "Virtual Head Robot Tracking and Three-Point l-l Control for Multiple Mobile Robots" *Proc. IEEE Workshop on Distributed Intelligent Systems*, Prague, Czech Republic, pp. 73-78, 14-16 June, 2006.

[Nguyen *et al.*, 2006c] Nguyen, A. D., Kwok, N. M., Ngo, V. T., and Ha, Q. P. "Collision-Free Formation with Reactively-Controlled Virtual Head Robot Tracking" *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, pp. 2509-2514, 9-15 October, 2006.

[Reeds and Shepp, 1990] Reeds, J. A., and Shepp, L. A., 1990 "Optimal paths for a car that goes both forwards and backwards" *Pacific Journal of Mathematics*, 145(2), pp. 367-393.

[Reif and Wang, 1999] Reif, J. H., and Wang, H., 1999 "Social potential fields: A distributed behavioral control for autonomous robots" *Robotics and Autonomous Systems*, 27(3), pp. 171-194.

[Ren and Beard, 2004] Ren, W., and Beard, R. W., 2004 "Decentralized Scheme for Spacecraft Formation Flying via the Virtual Structure Approach" *Journal of Guidance, Control, and Dynamics*, 27(1), pp. 73-82.

[Sheikholeslam and Desoer, 1992] Sheikholeslam, S., and Desoer, C. A., 1992 "Control of interconnected nonlinear dynamic systems: The platoon problem" *IEEE Transaction on Automatic Control*, 37(6), pp. 806-810.

[Siegwart and Nourbakhsh, 2004] Siegwart, R., and Nourbakhsh, I., 2004 *Introduction to autonomous mobile robots*, MIT Press, Cambridge, England.

[Skjetne *et al.*, 2003] Skjetne, R., Ihle, I.-A. F., and Fossen, T. I. "Formation control by Synchronizing Multiple Maneuvering Systems" *Proc. IFAC conference on Manoeuvring and Control of Marine Crafts*, Girona, Spain, pp. 280-285, 17-19 September, 2003.

[Skjetne *et al.*, 2002] Skjetne, R., Moi, S., and Fossen, T. I. "Nonlinear formation control of marine craft" *Proc. IEEE Conference on Decision and Control*, Las VeGas, Nevada, USA, pp. 1699-1704, 10-13 December, 2002.

[Spry and Hedrick, 2004] Spry, S., and Hedrick, J. K. "Formation control using generalized coordinates" *Proc. IEEE Conference on Decision and Control*, Atlantics, Paradise Island, Bahamas, pp. 2441-2446, 14-17 December, 2004.

[Stilwell, 2002] Stilwell, D. J. "Decentralized Control Synthesis for a Platoon of Autonomous Vehicles" *Proc. IEEE International Conference on Robotics & Automation*, Washington, DC, USA, pp. 744-749, 11-15 May, 2002.

[Stilwell and Bishop, 2000] Stilwell, D. J., and Bishop, B. E. "A Framework for Decentralized Control of Autonomous Vehicles" *Proc. IEEE International Conference on Robotics & Automation*, San Francisco, CA, USA, pp. 2358-2363, 24-28 April, 2000.

[Sugar and Kumar, 1998] Sugar, T., and Kumar, V. "Decentralized control of cooperating manipulators" *Proc. IEEE International Conference on Robotics and Automation*, Leuven, Belgium, pp. 2916-2921, 16-20 May, 1998.

[Sugihara and Suzuki, 1990] Sugihara, K., and Suzuki, I. "Distributed motion coordination of multiple mobile robots" *Proc. IEEE International Symposium on Intelligent Control*, Philadelphia, PA, USA, pp. 138-143, 5-7 September, 1990.

[Sugihara and Suzuki, 1996] Sugihara, K., and Suzuki, I., 1996 "Distributed algorithms for formation of geometric patterns with many mobile robots" *Journal of Robotic Systems*, 13(3), pp. 127-139.

[Suzuki and Yamashita, 1994] Suzuki, I., and Yamashita, M., 1994 "A theory of distributed anonymous mobile robots - Formation and agreement problems" *Technical Report TR-94-07-01*, Department of Electrical Engineering and Computer Science, University of Wisconsin, Milwaukee, USA.

[Takahashi *et al.*, 2004] 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), pp. 1272-1279.

[Tan and Lewis, 1996] Tan, K. H., and Lewis, M. A. "Virtual structures for high-precision cooperative mobile robotic control" *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Osaka, Japan, pp. 132-139, 04-08 November, 1996.

[Tanner *et al.*, 2003a] Tanner, H. G., Jadbabaie, A., and Pappas, G. J. "Stable flocking of mobile agents part II: dynamic topology" *Proc. EEE Conference on Decision and Control*, Maui, Hawaii, USA, pp. 2016-2021, 9-12 December, 2003.

[Tanner *et al.*, 2003b] Tanner, H. G., Jadbabaie, A., and Pappas, G. J. "Stable flocking of mobile agents, part I: fixed topology" *Proc. IEEE Conference on Decision and Control*, Maui, Hawaii, USA, pp. 2010-2015, 9-12 December, 2003.

[Tanner *et al.*, 2004] Tanner, H. G., Pappas, G. J., and Kumar, V., 2004 "Leader-to-Formation Stability" *IEEE Transaction on Robotics and Automation*, 20(3), pp. 443-455.

[Vidal *et al.*, 2003] Vidal, R., Shakernia, O., and Sastry, S. "Formation control of non-holonomic mobile robots with omnidirectional visual servoing and motion

segmentation" *Proc. IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, pp. 584-589, 14-19 September, 2003.

[Wang, 1991] Wang, P. K. C., 1991 "Navigation Strategies For Multiple Autonomous Mobile Robots Moving In Formation" *Journal of Robotic Systems*, 8(2), pp. 177-195.

[Wang and Davison, 1973] Wang, S., and Davison, E., 1973 "On the stabilisation of decentralised control systems" *IEEE Transaction on Automatic Control*, 18(5), pp. 473-478.

[Wang and Hadaegh, 1996] Wang, P. K. C., and Hadaegh, F. Y., 1996 "Coordination and control of multiple spacecraft moving in formation" *Journal of the Astronautical Sciences*, 44(3), pp. 315-355.

[Wang *et al.*, 1999] Wang, P. K. C., Hadaegh, F. Y., and Lau, K., 1999 "Synchronized Formation Rotation and Attitude Control of Multiple Free-Flying Spacecraft" *Journal of Guidance, Control, and Dynamics*, 22(1), pp. 28-35.

[Weimerskirch *et al.*, 2001] Weimerskirch, H., Martin, J., Clerquin, Y., Alexandre, P., and Jiraskova, S., 2001 "Energy saving in flight formations" *Nature*(413), pp. 697-698.

[Yamaguchi and Arai, 1994] Yamaguchi, H., and Arai, T. "Distributed and autonomous control method for generating shape of multiple mobile robot group" *Proc. IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, Munich, Germany, pp. 800-807, 12-16 September, 1994.

[Yamaguchi *et al.*, 2001] Yamaguchi, H., Arai, T., and Beni, G., 2001 "A distributed control scheme for multiple robotic vehicles to make group formations" *Robotics and Autonomous Systems*, 36(4), pp. 125-147.

[Yamaguchi and Burdick, 1998] Yamaguchi, H., and Burdick, J. W. "Asymptotic stabilization of multiple non-holonomic mobile robots forming group formations" *Proc. IEEE International Conference on Robotics anh Automation*, Leuven, Belgium, pp. 3573-3580, 16-20 May, 1998.

[Yamakita and Saito, 2004] Yamakita, M., and Saito, M. "Formation control of SMC with multiple coordinate systems" *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, pp. 1023-1028, 28 September - 2 October, 2004.

[Yan and Bitmead, 2003] Yan, J., and Bitmead, R. R. "Coordinated control and information architecture" *Proc. IEEE Conference on Decision and Control*, Maui, Hawaii, USA, pp. 3919-3923, 9-12 December, 2003.

[Yun *et al.*, 1997] Yun, X., Alptekin, G., and Albayrak, O., 1997 "Line and circle formation of distributed physical mobile robots" *Journal of Robotic Systems*, 14(2), pp. 63-76.

[Zak, 1989] Zak, M., 1989 "Terminal Attractor in Neural Networks" *Neural Networks*, 2(4), pp. 259-274.

[Zhang *et al.*, 2003] Zhang, F., Goldgeier, M., and Krishnaprasad, P. S. "Control of small formations using shape coordinates" *Proc. IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, pp. 2510-2515, 14-19 September, 2003.

Appendix A

AmigoBot - Specifications

Physical characteristics

Length	33 cm
Width	28 cm
Height (body)	13 cm
Body clearance	3 cm
Weight	3.6 Kg
Payload	1 Kg

Construction

Body	Molded polycarbonate
Chassis	1.6mm CNC fabricated aluminium
Assembly	Allen hex screws (metric)

Power

Battery	12V lead-acid
Charge	24.2 watt-hr
Run time	3+ hours

Recharge time (trickle)	8 hrs
Recharge time (fast)	3 hrs

Mobility

Drive wheels	2 solid rubber, with caster balance
Wheel diameter	10 cm
Wheel width	3 cm
Steering	Differential
Gear ratio	19.5:1
Swing radius	33 cm
Turn radius	0 cm
Translate speed max	750 mm/sec
Rotational speed max	300 degrees/sec
Traversable step max	1.5 cm
Traversable terrain	All wheelchair accessible

Sensors

Sonar	8 total
	1 each side
	4 forward
	2 rear
Position encoders	2 (one each motor)
	9,550 ticks per wheel revolution
	30 ticks per mm

Electronics

Processor	20 MHz Hitachi H8/2357
Position inputs	4
Sonar inputs	1 x 8 (multiplexed)
Digital I/O	6 digital IO logic ports
A/D	5 @ 0-5 VDC, 12-bit resolution
Digital timer inputs	6 @ 1μsec resolution
Comm port	3 RS-232 serial
FLASH	64 KB μP
	1M external
RAM	16 KB μP

Controls and ports

Main Power	Robot/accessories power ON/OFF
Charge	System power/battery recharge
RESET	Warm reboot/download
MOTORS/TEST	Motors/download/self-tests
Radio	Power and serial
Speaker	8-ohm
Serial ports	2 x RS232 (Control and System)
	1 TTL (AUX)

Appendix B

ARIA - Advanced Robotics Interface for Applications

B.1 ARIA overview

B.1.1 Introduction

Written in the C++ language, ARIA is client-side software for easy, high-performance access to and management of the robot server, as well as to the many accessory robot sensors and effectors. Its versatility and flexibility makes ARIA an excellent foundation for higher-level robotics applications.

ARIA can be run multi- or single-threaded, using its own wrapper around Linux pthreads and WIN32 threads. Use ARIA in many different ways, from simple command-control of the robot server for direct-drive navigation, to development of higher-level intelligent actions (behaviours).

New versions of ARIA can be downloaded from http://robots.activmedia.com/ARIA

B.1.2 ARIA-robot client-server relationship

The mobile robot servers, embodied in the Pioneer and AmigoBot Operating System software (ARCOS, AROS, P2OS, AmigOS, etc.) and found embedded on the robot's microcontroller, manage the low-level tasks of robot control and operation, including motion, heading and odometry, as well as acquiring sensor information (sonar and

compass, for example) and driving accessory components like the PTZ camera, TCM2 compass/inclinometer, and the Pioneer 5-DOF Arm. The robot servers do not, however, perform any high-level robotic tasks.

Rather, it is the job of an intelligent client running on a connected PC to perform the full gamut of robotics control strategies and tasks, such as obstacle detection and avoidance, sensor fusion, localization, features recognition, mapping, intelligent navigation, PTZ camera control, Arm motion, and much more. ARIA's role is on that intelligent client side.

Nearest the robot, ARIA's **ArDeviceConnection** class, at the behest of an application code, establishes and maintains a communication channel with the robot server, packaging commands to (**ArRobotPacketSender**) and decoding responses (**ArRobotPacketReceiver**) from the robot in safe and reliable packet formats (**ArRobotPacket**) prescribed by the client-server protocols.

At its heart, ARIA's **ArRobot** class collects and organizes the robot's operating states, and provides clear and convenient interface for other ARIA components, as well as upper-level applications, to access that robot state-reflection information for assessment, planning, and ultimately, intelligent, purposeful control of the platform and its accessories.

ArRobot's heart metaphor is particularly apt, too, since one of its important jobs is to maintain the clockwork cycles and multi-threaded rhythms of the robot-control system. Keyed to the robot's main information-packet cycle (hence, no longer a fixed timing cycle), **ArRobot**'s syncronous tasks (**ArSyncTask**) include the robot server-information packet handlers, sensor interpreters, action handlers, state reflectors, user tasks, and more. And a software may expand, replace, remove, and rearrange the list of synchronized tasks through **ArRobot**'s convenient sensor interp (**ArRobot::addSensorInterpTask**) and user task (**ArRobot::addUserTask**) related methods.

Through its Action class, ARIA provides a flexible, programmable mechanism for behaviour-level control of the robot server. An associated Resolver class lets one

organize and combine actions, for coordinated motion control and intelligent guidance. With ARIA actions, one easily develop integrated guarded-teleoperation and colour-blob tracking applications, for example.

ARIA also includes clear and convenient interface for applications to access and control ActivMedia Robotics accessory sensors and devices, including operation and state reflection for sonar and laser range finders, pan-tilt units, arms, inertial navigation devices, and many others.

B.1.3 Robot communication

One of the most important functions of ARIA, and one of the first and necessary things that an application must do, is to establish and manage client-server communications between an ARIA-based software client and the robot's onboard servers and devices.

Note that some accessories, such as the P2 Gripper, PTZ camera, P2 Arm, compass, and others, which attach to the robot's microcontroller AUX serial port, are controlled through the client-side device connection with the robot. Use different methods and procedures other than **ArDeviceConnection** to communicate with and manage those devices through ARIA

Connecting with a robot or the simulator - the easy way

One can use a convenience class called **ArSimpleConnector** to do the connection for him/her. **ArSimpleConnector** can also parse command line arguments so that one does not need to recompile to change where he/she want to connect. Among other benefits, an **ArSimpleConnector** will first try to connect to a simulator if one is running, otherwise it'll connect to a serial port... so one does not have to recompile his/her program for either mode, just don't have a simulator running, or have one running.

Connecting with a robot or the simulator - the hard way

ArDeviceConnection is ARIA's communications object; **ArSerialConnection** and **ArTcpConnection** are its built-in children most commonly used to manage communication between an ActivMedia robot or the robot simulator, respectively.

These classes are not device-specific, however, so use **ArSerialConnection**, for instance, to also configure a serial port and establish a connection with a robot accessory, such as with the SICK laser range finder.

Opening the connection

After creating and opening a device connection, associate it with its ARIA device handlers, most commonly with **ArRobot::setDeviceConnection** for the robot or the simulator.

For example, early in an ARIA program, specify the connection device and associate it with the robot:

```
ArTcpConnection con;
ArRobot robot;
```

Later in the program, after initializing the ARIA system (**Aria::init**(); is mandatory), set the Connection port to its default values (for TCP, host is "localhost" and port number is 8101), and then open the port:

```
con.setPort();
if (!con.openSimple())
{
        printf("Open failed.");
        Aria::shutdown();
        return 1;
}
```

TCP and Serial connections have their own implementation of open which is not inherited, but has default arguments that make the generic open work for the all default cases. And open returns a status integer which can be passed to the re-implemented and inherited **ArDeviceConnection::getOpenMessage** in order to retrieve related status string, which is useful in reporting errors to the user without having to know about the underlying device.

Robot client-server connection

After associating the device with the robot, now connect with the robot's servers, **ArRobot::blockingConnect** or **ArRobot::asyncConnect**, for example, to establish the client-server connection between ARIA **ArRobot** and the ActivMedia robot microcontroller or robot simulator. The blockingConnect method doesn't return from the call until a connection succeeds or fails:

```
robot.setDeviceConnection(&con);
if (!robot.blockingConnect())
{
        printf("Could not connect to robot... Exiting.");
        Aria::shutdown();
        return 1;
}
```

The previous examples connect with the simulator through a TCP socket on a PC. Use tcpConn.setPort(host, port) to set the TCP hostname or IP address and related socket number to another machine on the network. For instance, use tcpConn.setPort("bill", 8101); to connect to the simulator which is running on the networked computer "bill" through port 8101.

Replace **ArTcpConnection** con; with **ArSerialConnection** con; to connect with a robot through the default serial port (/dev/ttyS0 or COM1), or another one specify with **ArSerialConnection::setPort**(), such as con.setPort("COM3");.

At some point, one may want to open the port with the more verbose con.open().

Connection read, write, close and timestamping

The two main functions of a device connection are **ArDeviceConnection::read** and **ArDeviceConnection::write**. **ArDeviceConnection::close** also is inherited and important. One probably won't use direct read or write to the robot device, although he could. Rather, **ArRobot** provides a host of convenient methods that package his/her robot commands, and gather and distribute the various robot information packets, so that he/she does not have to attend those mundane details.

All **ArDeviceConnection** subclasses have support for timestamping (**ArDeviceConnection::getTimeRead**). With the robot connection, timestamping

merely says what time a robot SIP (Server Information Packet) came in, which can be useful for interpolating the robot's location more precisely.

B.1.4 ArRobot

As mentioned earlier, **ArRobot** is the heart of ARIA, acting as client-server communications gateway, central database for collection and distribution of state-reflection information, and systems synchronization manager. **ArRobot** is also the gathering point for many other robot tasks, including syncTasks, callbacks, range-finding sensor and Actions classes.

Client commands and server information packets

Client-server communications between applications software and an ActivMedia robot or the Simulator must adhere to strict packet-based protocols (in this context, the client is the software using ARIA to operate a robot, and the server is the robot's microcontroller.) **ArRobot** handles the low-level details of constructing and sending client-command packets to the robot as well as receiving and decoding the various Server Information Packets from the robot.

Packet handlers

Server Information Packets (SIPs) come from the robot over the robot-device connection and contain operating information about the robot and its accessories. Currently, there are two types of SIPs: the standard SIP and extended SIPs. The standard SIP gets sent by the robot to a connected client automatically every 100 (default) or 50 milliseconds. It contains the robot's current position, heading, translational and rotational speeds, freshly accumulated sonar readings, and much more. These data ultimately are stored and distributed by **ArRobot**'s State Reflection

Extended SIPs use the same communication-packet protocols as the standard SIP, but with a different "type" specification and, of course, containing different operating information, such as I/O port readings or accessory device states like for the Gripper. And, whereas the standard SIP gets sent automatically once per cycle, a client controls

extended packet communications by explicitly requesting that the server send one or more extended SIPs.

ArRobot's standard SIP handler automatically runs as an **ArRobot** synchronized task. Other SIP handlers are built in, but a client must add each to the connected robot object, and hence to the SIP handler sync task list, for it to take effect.

One also may add his/her own SIP handler with **ArRobot::addPacketHandler**. **ArListPos** keeps track of the order by which **ArRobot** calls each handler. When run, a packet handler must test the SIP type (**ArRobotPacket::getID**) and return true after decoding the packet type or return false, leaving the packet untouched for other handlers.

Command packets

From the client side going to the robot server, an ARIA program may send commands directly, or more commonly, use ARIA's convenience methods (Motion Commands and others) as well as engage Actions which ARIA ultimately converts into Direct Commands to the robot. At the ARIA-robot interface, there is no difference between Action- or other ARIA convenience-generated commands and Direct Commands. However, upper-level processes aren't necessarily aware of extraneous Direct or Motion Commands a client may send to the robot. Motion Commands in particular need special attention when mixing with Actions.

Once connected, an ARIA client may send commands to the robot server nearly at will, only limited by communication speeds and other temporal processes and delays. Similarly, the server responds nearly immediately with a requested SIP, such as a GRIPPERpac or IOpac which describe the P2 Gripper or Input/Output port states, respectively.

However, general information from the robot server about its odometry, current sonar readings, and the many other details which comprise its "standard" SIP automatically get sent to the ARIA client on a constant 100 or 50 millisecond cycle. This requires some synchronization with **ArRobot**.

Robot-ARIA synchronization

ArRobot runs a processing cycle: a series of synchronized tasks, including SIP handling, sensor interpretation, action handling and resolution, state reflection, and user tasks, in that order. By default, **ArRobot** performs these sequenced tasks each time it receives a standard SIP from the robot. Its cycle is thereby triggered by the robot so that the tasks get the freshest information from the robot upon which to act.

To begin **ArRobot**'s processing cycle, call **ArRobot::run**() to enter the cycle synchronously, or **ArRobot::runAsync**() to run the cycle in a new background thread. **ArRobot::stopRunning**() stops the processing cycle.

Of course, syncTasks runs without a connection with a robot, too. It has its own default cycle time of 100 milliseconds which one may examine and reset with **ArRobot::getCycleTime** and **ArRobot::setCycleTime**, respectively. **ArRobot** waits up to twice that cycle time for a standard SIP before cycling automatically.

ArRobot's synchronization task list is actually a tree, with five major branches. If a particular task is not running, none of its children will be called. Each task has an associated state value and a pointer to an **ArTaskState::State** variable, which can be used to control the process, by turning it on or off, or to see if it succeeded or failed. If the pointer is NULL, then it is assumed that the task does not care about its state, and a local variable will be used in the task structure to keep track of that tasks state.

For each branch, tasks get executed in descending order of priority.

ARIA provides convenient methods to add one's own sensor-interpretation and user tasks. Create an ARIA function pointer (Functors) and then add his/her sensor interpreter (**ArRobot::addSensorInterpTask**) or user task (**ArRobot::addUserTask**) to the list of syncTasks. These tasks can be removed; use **ArRobot::remSensorInterpTask** or **ArRobot::remUserTask** to remove sensor interpreter or user tasks, respectively, by name or by functor.

The intrepid ARIA programmer can add or prune branches from the **ArRobot** task list, as well as leaves on the branches. Do these things by getting the root of the tree with

ArRobot::getSyncTaskRoot, and then using the **ArSyncTask** class to do the desired manipulation.

One may disassociate **ArRobot**'s syncTask from firing when the standard SIP is received, through **ArRobot::setCycleChained**. But in doing so, he/she may degrade robot performance, as the robot's cycle will simply be run once every **ArRobot::getCycleTime** milliseconds.

State reflection

State reflection in the **ArRobot** class is the way ARIA maintains and distributes a snapshot of the robot's operating conditions and values, as extracted from the latest standard SIP.

The standard SIP also contains low-level sonar readings, which are reflected in **ArRobot** and examined with the methods: **ArRobot::getNumSonar**, **ArRobot::getSonarRange**, **ArRobot::isSonarNew**, **ArRobot::getSonarReading**, **ArRobot::getClosestSonarRange**, **ArRobot::getClosestSonarNumber**. This information is more useful when applied to a range device.

ARIA's **ArRobot** also, by default, reflects in the State Reflection Robot-ARIA Synchronization syncTask the latest client Motion Command to the robot server at a rate set by **ArRobot::setStateReflectionRefreshTime**. If no command is in effect, the **ArCommands::PULSE** Direct Command gets sent. State reflection of the motion command ensures that the client-server communication watchdog on the robot won't time out and disable the robot.

One may turn the motion-control state reflector off in the **ArRobot::ArRobot** constructor (set doStateReflection parameter to false). This will cause Motion Commands to be sent directly to the robot whenever they are called. State Reflection will send a PULSE command to the robot at **ArRobot::getStateReflectionRefreshTime** milliseconds to prevent the watchdog from timing out.

B.1.5 Range devices

Range devices (**ArRangeDevice**) are abstractions of sensors for which there are histories of relevant readings. Currently, there are two ARIA RangeDevices: sonar (**ArSonarDevice**) and the SICK laser (**ArSick**). All range devices are range-finding devices that periodically collect 2-D data at specific global coordinates, so the RangeDevice class should work for any type of two-dimensional sensor.

Attach a RangeDevice to a robot with **ArRobot::addRangeDevice** and remove it with **ArRobot::remRangeDevice**. Query for RangeDevices with **ArRobot::findRangeDevice**. **ArRobot::hasRangeDevice** will check to see if a particular range device (the given instance) is attached to the robot. A list of range devices can be obtained with **ArRobot::getRangeDeviceList**.

Note that sonar are integrated with the robot controller and that their readings automatically come included with the standard SIP and so are handled by the standard **ArRobot** packet handler. Nonetheless, one must explicitly add the sonar RangeDevice with his/her robot object to use the sonar readings for control tasks. ARIA's design gives the programmer ultimate control over their code, even though that means making him/her do nearly everything explicitly. Besides, not every program needs to track sonar data and there are some robots don't even have sonar.

Each RangeDevice has two sets of buffers (**ArRangeBuffer**): current and cumulative, and each support two different reading formats: box and polar. The current buffer contains the most recent reading; the cumulative buffer contains several readings over time, limited by **ArRangeBuffer::setSize**.

Useful for collision avoidance and other object detection tasks, apply the checkRangeDevices methods to conveniently scan a related buffer on all range devices attached to the robot for readings that fall within a specified range.

Note that each range device also has a threading mutex (**ArRangeDevice::lockDevice** and **ArRangeDevice::unlockDevice**) associated with it, so that sensors can be used in a thread-safe manner. For example, if a laser device gets added that runs in its own

thread, the checkRangeDevice functions on the robot lock the device so it can poke at the laser device without running into any issues, unlocking the device when it is done.

B.1.6 Commands and actions

An ARIA client drives the robot and runs its various accessories through Direct and Motion Commands, as well as through Actions.

Direct commands

At the very lowest level, one may send commands directly to the robot server through **ArRobot**. Direct commands consist of a 1-byte command number followed by none or more arguments, as defined by the robot's operating system (ARCOS, AROS, P2OS, AmigOS, etc.). For example, the command number 4 (ENABLE) enables the robot's motors if accompanied by the argument 1, and disables the motors with the argument 0.

Direct commands to the robot come in five flavours, each defined by its command argument type and length: use **ArRobot::com** for commands that have no argument, such as PULSE; **ArRobot::comInt** for a 2-byte integer argument, signed or unsigned, such as the motors ENABLE command; **ArRobot::com2Bytes** for when one want to define each of the two bytes in the argument, such as the VEL2 command; and **ArRobot::comStr** or **ArRobot::comStrN** for a null-terminated or defined-length (N extra argument) string argument, respectively, such as the sonar POLLING sequencing command.

The **ArCommands** class contains an enum with all the direct commands; **ArCommands::ENABLE**, for example. Although identical in syntax and effect when supported, not all Direct Commands are included with every ActivMedia robot. Fortunately, unrecognized or otherwise malformed client commands are benign since they get ignored by the server.

Motion commands

At a level just above **ArRobot**'s Direct Commands are the Motion Commands. These are explicit movement commands. Some have identical Direct Command analogues

and act to immediately control the mobility of a robot, either to set individual-wheel, or coordinated translational and rotational velocities (**ArRobot::setVel2**, **ArRobot::setVel**, **ArRobot::setRotVel**, respectively); change the robot's absolute or relative heading (**ArRobot::setHeading** or **ArRobot::setDeltaHeading**, respectively); move a prescribed distance (**ArRobot::move**); or just stop (**ArRobot::stop**).

Be aware that a Direct or a Motion Command may conflict with controls from Actions or other upper-level processes and lead to unexpected consequences. Use **ArRobot::clearDirectMotion** to cancel the overriding effect of a Motion Command so that an Action is able to regain control the robot. Or limit the time a Motion Command prevents other motion actions with **ArRobot::setDirectMotionPrecedenceTime**. Otherwise, the Motion Command will prevent actions forever. Use **ArRobot::getDirectMotionPrecedenceTime** to see how long a Motion Command takes precedence.

Actions

The best way to do non-trivial motion from ARIA is with its higher-level "Actions" mechanism. Actions are individual objects that independently provide motion requests which are evaluated and then combined each cycle to produce a final desired movement. This allows one to build complex behaviour from simple building blocks.

Actions are defined by creating a subclass of the **ArAction** the base class. **ArAction::fire** is the only function that needs to be overloaded for an action to work. ARIA includes a number of action classes.

Actions are added to robots with **ArRobot::addAction**, including a priority which determines its position in the action list. **ArAction::setRobot** is called on an action when it is added to a robot. One can override this. For example, this would be useful to add a connection callback, if there were some calculations he/she wished to do upon connection to the robot.

Actions are evaluated by the resolver in descending order of priority (lowest priority goes last) in each ArRobot syncTask cycle just prior to State Reflection. An action resolver goes through the actions to find a single end actionDesired (an

ArActionDesired object). Depending on its current state, each action contributes particular actionDesired movement values and strengths to the final action desired. After this final action desired has been calculated, it is stored and later gets passed to the State Reflector and on to the robot as motion commands.

As the resolver is evaluating each action it passes in the current action desired of the higher priority actions, this is the currentDesired. For example, a stall-recovery action could be programmed not to exert its motion effects if it has been pre-empted by a stop action, so this stall-recovery action would check and see if either the "strength" is "used up" or if there is a maximum velocity, and if so it can reset its state. However, there is no need for an action to pay attention to the currentDesired if not necessary (a resolver could also simply pass a **ArActionDesired.reset**() to the actions if it did not want the actions to know about its state.)

Action desired

ArActionDesired is the meat of actions. **ArActionDesired** objects should be reset (**ArActionDesired::reset**()) before they are used or reused.

There are six action channels:

- velocity (**ArActionDesired::setVel**),
- relative heading (**ArActionDesired::setDeltaHeading**),
- absolute heading (**ArActionDesired::setHeading**),
- maximum forward translational velocity (**ArActionDesired::setMaxVel**),
- maximum reverse translational velocity (**ArActionDesired::setMaxNegVel**), and
- maximum rotational velocity (**ArActionDesired::setMaxRotVel**).

An action gives each channel a strength between 0.0, the lowest, and 1.0, the highest. Strengths are used by the resolver to compute the relative effect the actionDesired

channel setting will have on the current translational velocity and heading of the robot, as well as the speed limits for those movements. Note that deltaHeading and heading are treated as the same channel for strength purposes, and that these are simply alternate ways of accessing the same channel.

The maximum velocity, maximum negative velocity, and maximum rotational velocity channels simply impose speed limits and thereby indirectly control the robot.

For more advanced usage, desired actions can be merged (**ArActionDesired::merge**) and averaged (**ArActionDesired::startAverage**, **ArActionDesired::addAverage**, **ArActionDesired::endAverage**).

Resolvers

ArResolver is the base action-resolver class. **ArPriorityResolver** is the default resolver. **ArResolver::resolve** is the function that **ArRobot** calls with the action list (actually **ArResolver::ActionMap**) in order to combine and thereby resolve the actionDesired movement controls into State Reflection motion commands to the robot server.

There may only be one resolver per robot, which is set with **ArRobot::setResolver**. However, a resolver could be created to contain within it multiple resolvers of its own. Note that although a robot has one particular resolver bound to it, a resolver instance is not tied to any robot. Thus, if one define a custom resolver, he/she could use one instance to work for all robots in a program.

The resolver works by setting each of the currentDesired channels to the contributing actionDesired values in proportion to their respective strenghts and priority, adjusting each movement channel's currentDesired value until the individual strength becomes 1.0 or the list is exhausted. Same-priority actions get averaged together (if they are competing) before being resolved with higher-priority results.

Predefined movement and limiting actions

For programming convenience, ARIA has defined two useful types of actions: Movement and Limiting. There are no classes for limiting or movement actions.

Built in movement actions have an **ArAction** prefix and act to set either or both the translational velocity (**setVel**) and heading (**setDeltaHeading** and **setHeading**) channels. Built in limiting actions are prefixed with **ArActionLimiter** and act to set one or more of the maximum translational and rotational velocity channels.

B.2 Aria compound list

Here are the classes, structs, unions and interfaces with brief descriptions:

ArAction Action class, what typically makes the robot move

ArActionAvoidFront This action does obstacle avoidance, controlling both trans and rot

ArActionAvoidSide Action to avoid impacts by firening into walls at a shallow angle

ArActionBumpers Action to deal with if the bumpers trigger

ArActionColorFollow ArActionColorFollow is an action that moves the robot toward the largest blob that appears in it's current field of view

ArActionConstantVelocity Action for going straight at a constant velocity

ArActionDeceleratingLimiter Action to limit the forwards motion of the robot based on range sensor readings

ArActionDesired Class used to say what movement is desired

ArActionDesiredChannel Class used by ArActionDesired for each channel, internal

ArActionGoto This action goes to a given ArPose very naively

ArActionGotoStraight This action goes to a given ArPose very naively

ArActionGroup Class for groups of actions to accomplish one thing

ArActionGroupColorFollow Follows a blob of colour

ArActionGroupInput Input to drive the robot

ArActionGroupRatioDrive Input to drive the robot

ArActionGroupRatioDriveUnsafe Input to drive the robot unsafely

ArActionGroupStop Stop the robot

ArActionGroupTeleop Teleop the robot

ArActionGroupUnguardedTeleop Teleop the robot in an unguarded and unsafe manner

ArActionGroupWander Has the robot wander

ArActionInput Action for taking input from outside to control the robot

ArActionIRs Action to deal with if the IRs trigger

ArActionJoydrive This action will use the joystick for input to drive the robot

ArActionKeydrive This action will use the keyboard arrow keys for input to drive the robot

ArActionLimiterBackwards Action to limit the backwards motion of the robot based on range sensor readings

ArActionLimiterForwards Action to limit the forwards motion of the robot based on range sensor readings

ArActionLimiterTableSensor Action to limit speed (and stop) based on whether the "table"-sensors see anything

ArActionMovementParameters This is a class for setting max velocities and accels and deciles

ArActionRatioInput Takes input on how to drive a robot in a more speed independent manner

ArActionRobotJoydrive This action will use the joystick for input to drive the robot

ArActionStallRecover Action to recover from a stall

ArActionStop Action for stopping the robot

ArActionTriangleDriveTo Action to drive up to a triangle found from an ArLineFinder

ArActionTurn Action to turn when the behaviors with more priority have limited the speed

ArACTS_1_2 Driver for ACTS

ArACTSBlob A class for the acts blob

ArAMPTU Driver for the AMPUT

ArAMPTUCommands A class with the commands for the AMPTU

ArAMPTUPacket A class for for making commands to send to the AMPTU

ArAnalogGyro Gyro plugin for the analog devices gyro

ArArg Argument class, mostly for actions, could be used for other things

ArArgumentBuilder This class is to build arguments for things that require argc and argv

ArArgumentParser Class for parsing arguments

ArASyncTask Asynchronous task (runs in its own thread)

ArBasePacket Base packet class

ArBumpers A class that treats the robot's bumpers as a range device

ArColor A class for holding colour information for ArDrawingData

ArCommands A class with an enum of the commands that can be sent to the robot

ArCondition Threading condition wrapper class

ArConfig Classes dealing with config files can inherit from this one

ArConfigArg Argument class for ArConfig

ArConfigGroup Container for holding a group of ArConfigs

ArConfigSection

ArConstFunctor1C< T, P1 > Functor for a const member function with 1 parameter

ArConstFunctor2C< T, P1, P2 > Functor for a const member function with 2 parameters

ArConstFunctor3C< T, P1, P2, P3 > Functor for a const member function with 3 parameters

ArConstFunctor4C< T, P1, P2, P3, P4 > Functor for a const member function with 4 parameters

ArConstFunctorC< T > Swig doesn't like the const functors, ArFunctors for const member functions, Functor for a const member function

ArConstRetFunctor1C< Ret, T, P1 > Functor for a const member function with return value and 1 parameter

ArConstRetFunctor2C< Ret, T, P1, P2 > Functor for a const member function with return value and 2 parameters

ArConstRetFunctor3C< Ret, T, P1, P2, P3 > Functor for a const member function with return value and 3 parameters

ArConstRetFunctor4C< Ret, T, P1, P2, P3, P4 > Functor for a const member function with return value and 4 parameters

ArConstRetFunctorC< Ret, T > Functor for a const member function with return value

ArDaemonizer Class that (in linux) will run the program as a daemon (ie fork it)

ArActionTriangleDriveTo::Data

ArDataLogger This class will log data, but one have to use it through an ArConfig right now

ArDeviceConnection Base class for device connections

ArDPPTU Driver for the DPPTU

ArDPPTUCommands A class with the commands for the DPPTU

ArDPPTUPacket A class for for making commands to send to the DPPTU

ArDrawingData

ArFileParser Class for parsing files more easily

ArForbiddenRangeDevice Class that takes forbidden lines and turns them into range readings

ArFunctor Base class for functors

ArFunctor1< P1 > Base class for functors with 1 parameter

ArFunctor1C< T, P1 > Functor for a member function with 1 parameter

ArFunctor2< P1, P2 > Base class for functors with 2 parameters

ArFunctor2C< T, P1, P2 > Functor for a member function with 2 parameters

ArFunctor3< P1, P2, P3 > Base class for functors with 3 parameters

ArFunctor3C< T, P1, P2, P3 > Functor for a member function with 3 parameters

ArFunctor4< P1, P2, P3, P4 > Base class for functors with 4 parameters

ArFunctor4C< T, P1, P2, P3, P4 > Functor for a member function with 4 parameters

ArFunctorASyncTask This is like ArASyncTask, but instead of runThread it uses a functor to run

ArFunctorC< T > Functor for a member function

ArGlobalFunctor Functor for a global function with no parameters

ArGlobalFunctor1< P1 > Functor for a global function with 1 parameter

ArGlobalFunctor2< P1, P2 > Functor for a global function with 2 parameters

ArGlobalFunctor3< P1, P2, P3 > Functor for a global function with 3 parameters

ArGlobalFunctor4< P1, P2, P3, P4 > Functor for a global function with 4 parameters

ArGlobalRetFunctor< Ret > Functor for a global function with return value

ArGlobalRetFunctor1< Ret, P1 > Functor for a global function with 1 parameter and return value

ArGlobalRetFunctor2< Ret, P1, P2 > Functor for a global function with 2 parameters and return value

ArGlobalRetFunctor3< Ret, P1, P2, P3 > Functor for a global function with 2 parameters and return value

ArGlobalRetFunctor4< Ret, P1, P2, P3, P4 > Functor for a global function with 4 parameters and return value

ArGripper A class of convenience functions for using the gripper

ArGripperCommands A class with an enum of the commands for the gripper

Aria This class performs global initialization and deinitialization

ArInterpolation

ArIrrfDevice A class for connecting to a PB-9 and managing the resulting data

ArIRs A class that treats the robot's Infareds as a range device

ArSoundsQueue::Item

ItemComparator

ItemComparator_OnlyData

ItemComparator_PriorityLessThan

ItemComparator_TypeAndData

ItemComparator_WithType

ItemComparator_WithTypePriorityLessThan

ArJoyHandler Interfaces to a joystick

ArKeyHandler This class will read input from the keyboard

ArSimpleConnector::LaserData Class that holds information about the laser data

ArLine This is the class for a line to do some geometric manipulation

ArLineFinder This class finds lines out of any range device with raw readings (ArSick for instance)

ArLineFinderSegment Class for ArLineFinder to hold more info than an ArLineSegment

ArLineSegment This is the class for a line segment to do some geometric manipulation

ArListPos Has enum for position in list

ArLog Logging utility class

ArLogFileConnection For connecting through a log file

ArMap This is a class for maps made with ScanStudio and Mapper3

ArMapObject This is a class for objects within an ArMap

ArMath This class has static members to do common math operations

ArMode A class for different modes, mostly as related to keyboard input

ArModeActs Mode for following a colour blob using ACTS

ArModeCamera Mode for controlling the camera

ArModeGripper Mode for controlling the gripper

ArModeSonar Mode for displaying the sonar

ArModeTCM2 Mode for following a colour blob using ACTS

ArModeTeleop Mode for teleoping the robot with joystick + keyboard

ArModeUnguardedTeleop Mode for teleoping the robot with joystick + keyboard

ArModeWander Mode for wandering around

ArModule Dynamicly loaded module base class, read warning in more

ArModuleLoader Dynamic ArModule loader

ArMutex Mutex wrapper class

ArNetServer Class for running a simple net server to send/recv commands via text

ArP2Arm Arm Control class

P2ArmJoint P2 Arm joint info

ArPose The class which represents a position

ArPoseWithTime A subclass of pose that also has the time the pose was taken

ArPriority Has enum for priority (mostly for ArConfig)

ArPriorityResolver (Default resolver), takes the action list and uses the priority to resolve

ArConfig::ProcessFileCBType

ArPTZ Base class which handles the PTZ cameras

ArRangeBuffer This class is a buffer that holds ranging information

ArRangeDevice The class for all devices which return range info (laser, sonar)

ArRangeDeviceThreaded A range device which can run in its own thread

ArRatioInputJoydrive This action will use the joystick for input to drive the robot

ArRatioInputKeydrive This will use the keyboard arrow keys and the ArActionRatioInput to drive the robot

ArRatioInputRobotJoydrive This action will use the joystick on the robot to drive

ArRecurrentTask Recurrent task (runs in its own thread)

ArResolver Resolves a list of actions and returns what to do

ArRetFunctor< Ret > Base class for functors with a return value

ArRetFunctor1< Ret, P1 > Base class for functors with a return value with 1 parameter

ArRetFunctor1C< Ret, T, P1 > Functor for a member function with return value and 1 parameter

ArRetFunctor2< Ret, P1, P2 > Base class for functors with a return value with 2 parameters

ArRetFunctor2C< Ret, T, P1, P2 > Functor for a member function with return value and 2 parameters

ArRetFunctor3< Ret, P1, P2, P3 > Base class for functors with a return value with 3 parameters

ArRetFunctor3C< Ret, T, P1, P2, P3 > Functor for a member function with return value and 3 parameters

ArRetFunctor4< Ret, P1, P2, P3, P4 > Base class for functors with a return value with 4 parameters

ArRetFunctor4C< Ret, T, P1, P2, P3, P4 > Functor for a member function with return value and 4 parameters

ArRetFunctorC< Ret, T > Functor for a member function with return value

ArRingQueue< T >

ArRobot The important class

ArRobotConfigPacketReader This class will read a config packet from the robot

ArRobotJoyHandler Interfaces to a joystick on the robot's microcontroller

ArRobotPacket Represents the packets sent to the robot as well as those received from it

ArRobotPacketReceiver Given a device connection it receives packets from the robot through it

ArRobotPacketSender Given a device connection this sends commands through it to the robot

ArRobotParams Contains the robot parameters, according to the parameter file

ArRunningAverage This is a class for computing a running average of a number of elements

ArSectors A class for keeping track of if a complete revolution has been attained

ArSensorReading A class to hold a sensor reading, should be one instance per sensor

ArSerialConnection For connecting to devices through a serial port

ArSick The sick driver

ArSickLogger This class can be used to create log files for the laser mapper

ArSickPacket Represents the packets sent to the sick as well as those received from it

ArSickPacketReceiver Given a device connection it receives packets from the sick through it

ArSignalHandler Signal handling class

ArSimpleConnector This class simplifies connecting to the robot and/or laser

ArSocket Socket communication wrapper

ArSonarDevice A class for keeping track of sonar

ArSonyPacket A class for for making commands to send to the Sony

ArSonyPTZ A class to use the Sony pan tilt zoom unit

ArSoundPlayer This class provides a cross-platform interface for playing short sound samples (currently implemented for Windows and Linux)

ArSoundsQueue This class manages a queue of items to play as WAV files or as text to speak using a speech synthesizer

ArSpeechSynth Abstract interface to speech synthesis

ArStringInfoHolder This class holds information about strings (helper for other things)

ArStringInfoHolderFunctions This class just holds some helper functions for the ArStringInfoHolder

ArSyncTask Class used internally to manage the functions that are called every cycle

ArTaskState Class with the different states a task can be in

ArTCM2 This class will get extra information that the tcm2 supplies

ArTcpConnection For connection to a device through a socket

ArThread POSIX/WIN32 thread wrapper class

ArTime A class for time readings

ArTransform A class to handle transforms between different coordinates

ArTypes Contains platform independent sized variable types

ArUtil This class has utility functions

ArVCC4 Driver for the VCC4

ArVCC4Commands A class with the commands for the VCC4

ArVCC4Packet A class for making commands to send to the VCC4

ArVersalogicIO

Appendix C

Algorithm for Matlab-Simulink simulation

A proposed algorithm for Matlab-Simulink simulation is described in Figure C.1. The main Matlab program file (*.m* file) will prepare all initial parameters for robots in a group and call sub-program files with respect to steps (for example, program *step2.m* for step 2 of the formation initialisation procedure as presented in Chapter 5).

For example, the main program *formation51.m* for Simulation 5.1 in Chapter 5 is as follows.

```
% Formation control for 5 mobile robots
% Written by Anh Duy Nguyen - 1/2006

global collision;

% Initial conditions of robots
Xi1=30; Yi1=0; Oi1=0;
Xi2=0; Yi2=0; Oi2=0;
Xi3=10; Yi3=50; Oi3=0;
Xi4=60; Yi4=80; Oi4=2;
Xi5=30; Yi5=-130; Oi5=3;

% Speed of the Lead
vt1=[0 5]; va1=[0 0];

% Parameters for Tracking Control
R2=60; L2=0; l2=1; lamda21=1; lamda22=2;
R3=-60; L3=0; l3=1; lamda31=1; lamda32=2;
R4=60; L4=0; l4=1; lamda41=1; lamda42=2;
R5=-60; L5=0; l5=1; lamda51=1; lamda52=2;

% Parameters for Collision avoidance
Tr=[0 3];Dd=[0 22];collision=0;
```

```
%Begin

%  t: Final Simulation time
%  tr: Simulation time of each step
%  a: string variable to record the switching between VHRT-3PLL
t=0; tr=0;i=0; a='';

%  Final  states of robots
xyo1f=[]; xyo2f=[]; xyo3f=[];xyo4f=[];xyo5f=[];

while t<30
   step2;
end
while ((t>=30)&&(t<70))
   step3;
end
```

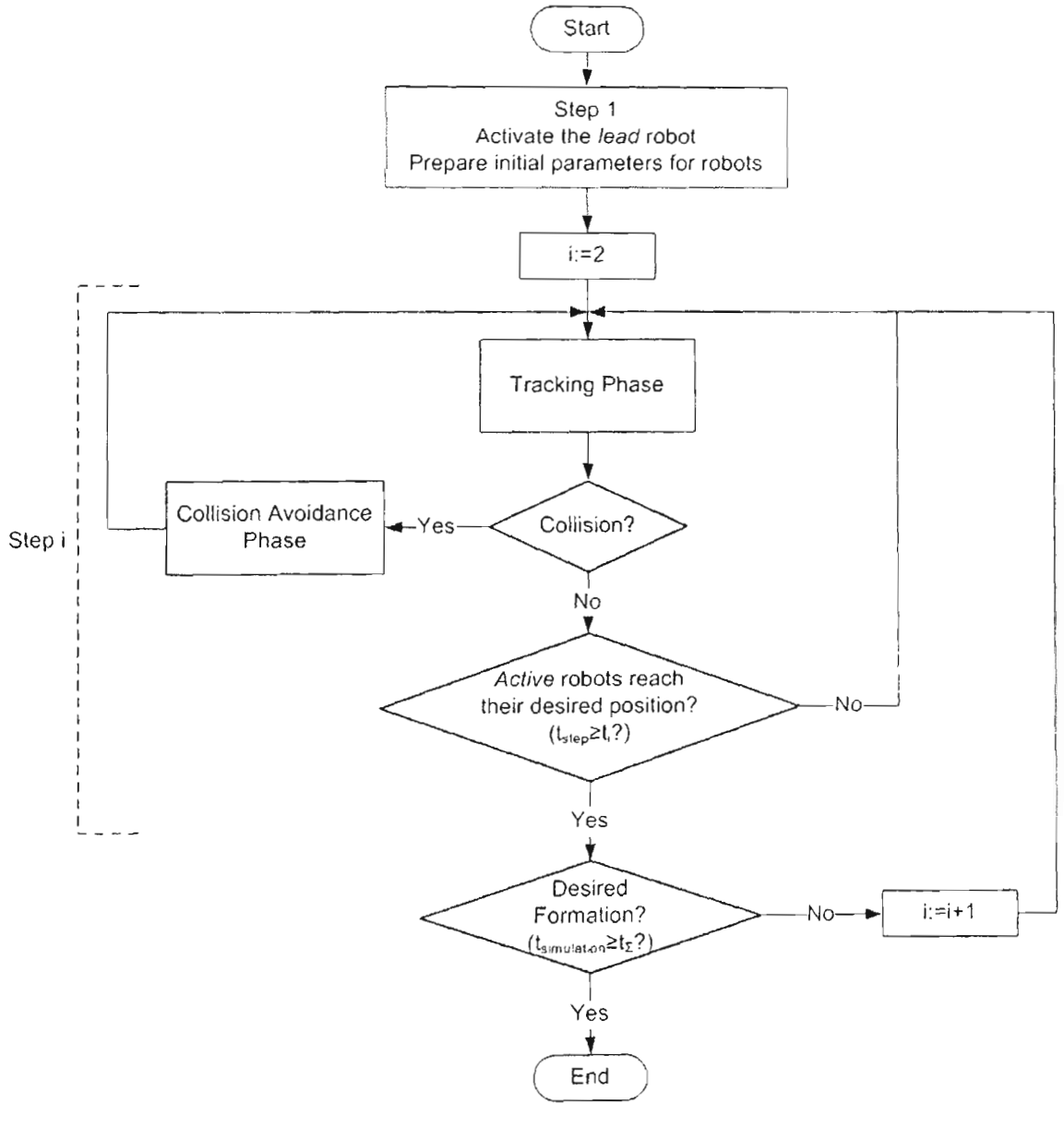


Figure C.1: Algorithm for simulation

formation51.m calls *step2.m* and *step 3.m*, which are written at the end of this appendix

Time limit for step i (t_i) is chosen large enough to ensure the part of desired formation can be established at the end of the step. For example, in the above simulation $t_2 = 30s$. Similarly, time limit for the whole process (t_Σ) should be chosen so that after tracking and collision avoidance phases in steps, the ultimate desired formation can be obtained. In this simulation $t_\Sigma = 70s$.

For the sake of simplicity, tracking and collision avoidance phases can be simulated using Simulink models. For example, *Step2_RL.mdl* for tracking phase at step 2, *Step312_4153.mdl* for collision avoidance between robot 1-robot 4 and robot 3-robot 5 (if necessary) at step 3 of Simulations 5.1 and 5.2 (Chapter 5). The immediate state of each robot at each step is recorded in an array variable (for example, *xyo1* for robot 1). The final states of robots for the whole initialisation process are combined in variables *xyoif* with $i = \overline{1,N}$ (e.g. *xyo1f* for robot 1).

The following scripts are the Matlab code of sub-programs *step2.m* and *step3.m* of the Simulation 5.1 in Chapter 5

step 2.m

```
D11=[0 5]; D12=[0 2.5]; r3=[0 2.5];
l13d=[0 60]; l23d=[0 90];
sim('Step2_RL');
   i=i+1;
   m=size(xyo1,1);
   tr(i)=xyo1(m,1);
   Xi1=xyo1(m,2); Yi1=xyo1(m,3); Oi1=xyo1(m,4);
   Xi2=xyo2(m,2); Yi2=xyo2(m,3); Oi2=xyo2(m,4);
   Xi3=xyo3(m,2); Yi3=xyo3(m,3); Oi3=xyo3(m,4);
   Xi4=xyo4(m,2); Yi4=xyo4(m,3); Oi4=xyo4(m,4);
   Xi5=xyo5(m,2); Yi5=xyo5(m,3); Oi5=xyo5(m,4);
   xyo1(:,1)=xyo1(:,1)+t;
   xyo2(:,1)=xyo2(:,1)+t;
   xyo3(:,1)=xyo3(:,1)+t;
   xyo4(:,1)=xyo4(:,1)+t;
   xyo5(:,1)=xyo5(:,1)+t;
   xyo1f=[xyo1f; xyo1];
   xyo2f=[xyo2f; xyo2];
   xyo3f=[xyo3f; xyo3];
   xyo4f=[xyo4f; xyo4];
   xyo5f=[xyo5f; xyo5];

   t=t+tr(i); a=[a ' T1'];
```

```
if tr(i)<30
   collision=1;
else
   collision=0;
end
ll12=1;
while collision==1                          %while 1
  while ((collision==1) && (ll12==1))       %while 2
     D=D11; r=[0 0];
     Xi3h=Xi3+r(2)*sin(Oi3)+D(2)*cos(Oi3);
     Yi3h=Yi3-r(2)*cos(Oi3)+D(2)*sin(Oi3);
     l130=sqrt((Xi1-Xi3h)*(Xi1-Xi3h)+(Yi1-Yi3h)*(Yi1-Yi3h));
     l230=sqrt((Xi2-Xi3h)*(Xi2-Xi3h)+(Yi2-Yi3h)*(Yi2-Yi3h));
     l130=[0 l130];
     l230=[0 l230];
     sim('Step2_LL12');
     i=i+1;
     m=size(xyo1,1);
     tr(i)=xyo1(m,1);
     Xi1=xyo1(m,2); Yi1=xyo1(m,3); Oi1=xyo1(m,4);
     Xi2=xyo2(m,2); Yi2=xyo2(m,3); Oi2=xyo2(m,4);
     Xi3=xyo3(m,2); Yi3=xyo3(m,3); Oi3=xyo3(m,4);
     xyo1(:,1)=xyo1(:,1)+t;
     xyo2(:,1)=xyo2(:,1)+t;
     xyo3(:,1)=xyo3(:,1)+t;
     xyo4(:,1)=xyo4(:,1)+t;
     xyo5(:,1)=xyo5(:,1)+t;
     xyo1f=[xyo1f;  xyo1];
     xyo2f=[xyo2f;  xyo2];
     xyo3f=[xyo3f;  xyo3];
     xyo4f=[xyo4f;  xyo4];
     xyo5f=[xyo5f;  xyo5];
     t=t+tr(i); a=[a ' C1'];
     if tr(i)<Tr(2)
        collision=1;
     else
        collision=0;
     end
     if ((collision==1) && (ll12==1))
        D=D12;r=[0 0];
        Xi3h=Xi3+r(2)*sin(Oi3)+D(2)*cos(Oi3);
        Yi3h=Yi3-r(2)*cos(Oi3)+D(2)*sin(Oi3);
        l130=sqrt((Xi1-Xi3h)*(Xi1-Xi3h)+(Yi1-Yi3h)*(Yi1-Yi3h));
        l230=sqrt((Xi2-Xi3h)*(Xi2-Xi3h)+(Yi2-Yi3h)*(Yi2-Yi3h));
        l130=[0 l130];
        l230=[0 l230];
        sim('Step2_LL12');
        i=i+1;
        m=size(xyo1,1);
        tr(i)=xyo1(m,1);
        Xi1=xyo1(m,2); Yi1=xyo1(m,3); Oi1=xyo1(m,4);
        Xi2=xyo2(m,2); Yi2=xyo2(m,3); Oi2=xyo2(m,4);
        Xi3=xyo3(m,2); Yi3=xyo3(m,3); Oi3=xyo3(m,4);
        Xi4=xyo4(m,2); Yi4=xyo4(m,3); Oi4=xyo4(m,4);
        Xi5=xyo5(m,2); Yi5=xyo5(m,3); Oi5=xyo5(m,4);
        xyo1(:,1)=xyo1(:,1)+t;
        xyo2(:,1)=xyo2(:,1)+t;
        xyo3(:,1)=xyo3(:,1)+t;
```

```
        xyo4(:,1)=xyo4(:,1)+t;
        xyo5(:,1)=xyo5(:,1)+t;
        xyo1f=[xyo1f;  xyo1];
        xyo2f=[xyo2f;  xyo2];
        xyo3f=[xyo3f;  xyo3];
        xyo4f=[xyo4f;  xyo4];
        xyo5f=[xyo5f;  xyo5];
        t=t+tr(i); a=[a ' C2'];
        if tr(i)<Tr(2)
        collision=1;
        else
        collision=0;
        end

        if ((tr(i)==0) && (tr(i-1)==0))
        ll12=0;
        end
      end
    end                                %while 2
    if ((collision==1) && (ll12==0))
      D=D11; r=r3;
      Xi3h=Xi3+r(2)*sin(Oi3)+D(2)*cos(Oi3);
      Yi3h=Yi3-r(2)*cos(Oi3)+D(2)*sin(Oi3);
      l130=sqrt((Xi1-Xi3h)*(Xi1-Xi3h)+(Yi1-Yi3h)*(Yi1-Yi3h));
      l230=sqrt((Xi2-Xi3h)*(Xi2-Xi3h)+(Yi2-Yi3h)*(Yi2-Yi3h));
      l130=[0 l130];
      l230=[0 l230];
      sim('Step2_LL3');
      i=i+1;
      m=size(xyo1,1);
      tr(i)=xyo1(m,1);
      Xi1=xyo1(m,2); Yi1=xyo1(m,3); Oi1=xyo1(m,4);
      Xi2=xyo2(m,2); Yi2=xyo2(m,3); Oi2=xyo2(m,4);
      Xi3=xyo3(m,2); Yi3=xyo3(m,3); Oi3=xyo3(m,4);
      Xi4=xyo4(m,2); Yi4=xyo4(m,3); Oi4=xyo4(m,4);
      Xi5=xyo5(m,2); Yi5=xyo5(m,3); Oi5=xyo5(m,4);
      xyo1(:,1)=xyo1(:,1)+t;
      xyo2(:,1)=xyo2(:,1)+t;
      xyo3(:,1)=xyo3(:,1)+t;
      xyo4(:,1)=xyo4(:,1)+t;
      xyo5(:,1)=xyo5(:,1)+t;
      xyo1f=[xyo1f;  xyo1];
      xyo2f=[xyo2f;  xyo2];
      xyo3f=[xyo3f;  xyo3];
      xyo4f=[xyo4f;  xyo4];
      xyo5f=[xyo5f;  xyo5];
      t=t+tr(i); a=[a ' C3'];
      if tr(i)<Tr(2)
      collision=1;
      else
      collision=0;
      end
      ll12=1;
    end
  end                                  % while 1
```

step3.m

```
D2=5;g=2;
c=0;
sim('Step312_RL');
   i=i+1;
   m=size(xyo1,1);
   tr(i)=xyo1(m,1);

   Xi1=xyo1(m,2); Yi1=xyo1(m,3); Oi1=xyo1(m,4);
   Xi2=xyo2(m,2); Yi2=xyo2(m,3); Oi2=xyo2(m,4);
   Xi3=xyo3(m,2); Yi3=xyo3(m,3); Oi3=xyo3(m,4);
   Xi4=xyo4(m,2); Yi4=xyo4(m,3); Oi4=xyo4(m,4);
   Xi5=xyo5(m,2); Yi5=xyo5(m,3); Oi5=xyo5(m,4);
   xyo1(:,1)=xyo1(:,1)+t;
   xyo2(:,1)=xyo2(:,1)+t;
   xyo3(:,1)=xyo3(:,1)+t;
   xyo4(:,1)=xyo4(:,1)+t;
   xyo5(:,1)=xyo5(:,1)+t;
   xyo1f=[xyo1f;  xyo1];
   xyo2f=[xyo2f;  xyo2];
   xyo3f=[xyo3f;  xyo3];
   xyo4f=[xyo4f;  xyo4];
   xyo5f=[xyo5f;  xyo5];

   t=t+tr(i); a=[a ' T2'];
   if tr(i)<40
      collision=1;
   else
      collision=0;
   end

while collision==1
      X4d=Xi2+R4*sin(Oi2)-L4*cos(Oi2);
      Y4d=Yi2-R4*cos(Oi2)-L4*sin(Oi2);
      Xi4h=Xi4+D2*cos(Oi4);
      Yi4h=Yi4+D2*sin(Oi4);

      X5d=Xi3+R5*sin(Oi3)-L5*cos(Oi3);
      Y5d=Yi3-R5*cos(Oi3)-L5*sin(Oi3);
      Xi5h=Xi5+D2*cos(Oi5);
      Yi5h=Yi5+D2*sin(Oi5);

      Rleader_a=Dd(2)+D2; Rleader_b=-Rleader_a;

      % Leader 2 for Robot1
      X1leader_a=Xi1+Rleader_a*sin(Oi1);
      Y1leader_a=Yi1-Rleader_a*cos(Oi1);
      X1leader_b=Xi1+Rleader_b*sin(Oi1);
      Y1leader_b=Yi1-Rleader_b*cos(Oi1);

      test4_a=(X1leader_a - X4d)*(X1leader_a - X4d)+(Y1leader_a - Y4d)*(Y1leader_a - Y4d);
      test4_b=(X1leader_b - X4d)*(X1leader_b - X4d)+(Y1leader_b - Y4d)*(Y1leader_b - Y4d);
      if test4_a >= test4_b
         R41leader2=Rleader_b;
         X41leader2=X1leader_b;
```

```
    Y41leader2=Y1leader_b;
else
    R41leader2=Rleader_a;
    X41leader2=X1leader_a;
    Y41leader2=Y1leader_a;
end

test5_a=(X1leader_a - X5d)*(X1leader_a - X5d)+(Y1leader_a - Y5d)*(Y1leader_a - Y5d);
test5_b=(X1leader_b - X5d)*(X1leader_b - X5d)+(Y1leader_b - Y5d)*(Y1leader_b - Y5d);
if test5_a >= test5_b
    R51leader2=Rleader_b;
    X51leader2=X1leader_b;
    Y51leader2=Y1leader_b;
else
    R51leader2=Rleader_a;
    X51leader2=X1leader_a;
    Y51leader2=Y1leader_a;
end

% Leader 2 for Robot2
X2leader_a=Xi2+Rleader_a*sin(Oi2);
Y2leader_a=Yi2-Rleader_a*cos(Oi2);
X2leader_b=Xi2+Rleader_b*sin(Oi2);
Y2leader_b=Yi2-Rleader_b*cos(Oi2);

test4_a=(X2leader_a - X4d)*(X2leader_a - X4d)+(Y2leader_a - Y4d)*(Y2leader_a - Y4d);
test4_b=(X2leader_b - X4d)*(X2leader_b - X4d)+(Y2leader_b - Y4d)*(Y2leader_b - Y4d);
if test4_a >= test4_b
    R42leader2=Rleader_b;
    X42leader2=X2leader_b;
    Y42leader2=Y2leader_b;
else
    R42leader2=Rleader_a;
    X42leader2=X2leader_a;
    Y42leader2=Y2leader_a;
end

test5_a=(X2leader_a - X5d)*(X2leader_a - X5d)+(Y2leader_a - Y5d)*(Y2leader_a - Y5d);
test5_b=(X2leader_b - X5d)*(X2leader_b - X5d)+(Y2leader_b - Y5d)*(Y2leader_b - Y5d);
if test5_a >= test5_b
    R52leader2=Rleader_b;
    X52leader2=X2leader_b;
    Y52leader2=Y2leader_b;
else
    R52leader2=Rleader_a;
    X52leader2=X2leader_a;
    Y52leader2=Y2leader_a;
end

% Leader 2 for Robot3
X3leader_a=Xi3+Rleader_a*sin(Oi3);
Y3leader_a=Yi3-Rleader_a*cos(Oi3);
X3leader_b=Xi3+Rleader_b*sin(Oi3);
Y3leader_b=Yi3-Rleader_b*cos(Oi3);

test4_a=(X3leader_a - X4d)*(X3leader_a - X4d)+(Y3leader_a - Y4d)*(Y3leader_a - Y4d);
test4_b=(X3leader_b - X4d)*(X3leader_b - X4d)+(Y3leader_b - Y4d)*(Y3leader_b - Y4d);
```

```
if test4_a >= test4_b
   R43leader2=Rleader_b;
   X43leader2=X3leader_b;
   Y43leader2=Y3leader_b;
else
   R43leader2=Rleader_a;
   X43leader2=X3leader_a;
   Y43leader2=Y3leader_a;
end

test5_a=(X3leader_a - X5d)*(X3leader_a - X5d)+(Y3leader_a - Y5d)*(Y3leader_a - Y5d);
test5_b=(X3leader_b - X5d)*(X3leader_b - X5d)+(Y3leader_b - Y5d)*(Y3leader_b - Y5d);
if test5_a >= test5_b
   R53leader2=Rleader_b;
   X53leader2=X3leader_b;
   Y53leader2=Y3leader_b;
else
   R53leader2=Rleader_a;
   X53leader2=X3leader_a;
   Y53leader2=Y3leader_a;
end

% Leader 2 for Robot4
X4leader_a=Xi4+Rleader_a*sin(Oi4);
Y4leader_a=Yi4-Rleader_a*cos(Oi4);
X4leader_b=Xi4+Rleader_b*sin(Oi4);
Y4leader_b=Yi4-Rleader_b*cos(Oi4);

test5_a=(X4leader_a - X5d)*(X4leader_a - X5d)+(Y4leader_a - Y5d)*(Y4leader_a - Y5d);
test5_b=(X4leader_b - X5d)*(X4leader_b - X5d)+(Y4leader_b - Y5d)*(Y4leader_b - Y5d);
if test5_a >= test5_b
   R54leader2=Rleader_b;
   X54leader2=X3leader_b;
   Y54leader2=Y3leader_b;
else
   R54leader2=Rleader_a;
   X54leader2=X4leader_a;
   Y54leader2=Y4leader_a;
end

% m=size(xyo1,1);
% tr(i)=xyo1(m,1);
if d41(m,1)<=Dd(2)  c=41;     end
if d42(m,1)<=Dd(2)  c=42;     end
if d43(m,1)<=Dd(2)  c=43;     end
if d51(m,1)<=Dd(2)  c=51;     end
if d52(m,1)<=Dd(2)  c=52;     end
if d53(m,1)<=Dd(2)  c=53;     end
if d54(m,1)<=Dd(2)  c=54;     end
if ((d41(m,1)<=Dd(2))&&(d51(m,1)<=Dd(2)))  c=4151;  end
if ((d41(m,1)<=Dd(2))&&(d52(m,1)<=Dd(2)))  c=4152;  end
if ((d41(m,1)<=Dd(2))&&(d53(m,1)<=Dd(2)))  c=4153;  end
if ((d41(m,1)<=Dd(2))&&(d54(m,1)<=Dd(2)))  c=4154;  end
if ((d42(m,1)<=Dd(2))&&(d51(m,1)<=Dd(2)))  c=4251;  end
if ((d42(m,1)<=Dd(2))&&(d52(m,1)<=Dd(2)))  c=4252;  end
if ((d42(m,1)<=Dd(2))&&(d53(m,1)<=Dd(2)))  c=4253;  end
if ((d42(m,1)<=Dd(2))&&(d54(m,1)<=Dd(2)))  c=4254;  end
```

```
    if ((d43(m,1)<=Dd(2))&&(d51(m,1)<=Dd(2)))   c=4351;     end
    if ((d43(m,1)<=Dd(2))&&(d52(m,1)<=Dd(2)))   c=4352;     end
    if ((d43(m,1)<=Dd(2))&&(d53(m,1)<=Dd(2)))   c=4353;     end
    if ((d43(m,1)<=Dd(2))&&(d54(m,1)<=Dd(2)))   c=4354;     end
    if ((d51(m,1)<=Dd(2))&&(d54(m,1)<=Dd(2)))   c=5154;     end
    if ((d52(m,1)<=Dd(2))&&(d54(m,1)<=Dd(2)))   c=5254;     end
    if ((d53(m,1)<=Dd(2))&&(d54(m,1)<=Dd(2)))   c=5354;     end
switch c
   case 41
    R4leader2=R41leader2;
    l4130=sqrt((Xi1-Xi4h)*(Xi1-Xi4h)+(Yi1-Yi4h)*(Yi1-Yi4h));
    l4230=sqrt((X41leader2-Xi4h)*(X41leader2-Xi4h)+(Y41leader2-Yi4h)*(Y41leader2-Yi4h));
    l413d=Dd(2)+D2+g;l423d=g;
    sim('Step312_41'); a=[a ' 41'];
   case 42
    R4leader2=R42leader2;
    l4130=sqrt((Xi2-Xi4h)*(Xi2-Xi4h)+(Yi2-Yi4h)*(Yi2-Yi4h));
    l4230=sqrt((X42leader2-Xi4h)*(X42leader2-Xi4h)+(Y42leader2-Yi4h)*(Y42leader2-Yi4h));
    l413d=Dd(2)+D2+g;l423d=g;
    sim('Step312_42'); a=[a ' 42'];
   case 43
    R4leader2=R43leader2;
    l4130=sqrt((Xi3-Xi4h)*(Xi3-Xi4h)+(Yi3-Yi4h)*(Yi3-Yi4h));
    l4230=sqrt((X43leader2-Xi4h)*(X43leader2-Xi4h)+(Y43leader2-Yi4h)*(Y43leader2-Yi4h));
    l413d=Dd(2)+D2+g;l423d=g;
    sim('Step312_43'); a=[a ' 43'];
   case 51
    R5leader2=R51leader2;
    l5130=sqrt((Xi1-Xi5h)*(Xi1-Xi5h)+(Yi1-Yi5h)*(Yi1-Yi5h));
    l5230=sqrt((X51leader2-Xi5h)*(X51leader2-Xi5h)+(Y51leader2-Yi5h)*(Y51leader2-Yi5h));
    l513d=Dd(2)+D2+g;l523d=g;
    sim('Step312_51'); a=[a ' 51'];
   case 52
    R5leader2=R52leader2;
    l5130=sqrt((Xi2-Xi5h)*(Xi2-Xi5h)+(Yi2-Yi5h)*(Yi2-Yi5h));
    l5230=sqrt((X52leader2-Xi5h)*(X52leader2-Xi5h)+(Y52leader2-Yi5h)*(Y52leader2-Yi5h));
    l513d=Dd(2)+D2+g;l523d=g;
    sim('Step312_52'); a=[a ' 52'];
   case 53
    R5leader2=R53leader2;
    l5130=sqrt((Xi3-Xi5h)*(Xi3-Xi5h)+(Yi3-Yi5h)*(Yi3-Yi5h));
    l5230=sqrt((X53leader2-Xi5h)*(X53leader2-Xi5h)+(Y53leader2-Yi5h)*(Y53leader2-Yi5h));
    l513d=Dd(2)+D2+g;l523d=g;
    sim('Step312_53'); a=[a ' 53'];
   case 54
    R5leader2=R54leader2;
    l5130=sqrt((Xi4-Xi5h)*(Xi4-Xi5h)+(Yi4-Yi5h)*(Yi4-Yi5h));
    l5230=sqrt((X54leader2-Xi5h)*(X54leader2-Xi5h)+(Y54leader2-Yi5h)*(Y54leader2-Yi5h));
    l513d=Dd(2)+D2+g;l523d=g;
    sim('Step312_54'); a=[a ' 54'];
   case 4151
    R4leader2=R41leader2;
    l4130=sqrt((Xi1-Xi4h)*(Xi1-Xi4h)+(Yi1-Yi4h)*(Yi1-Yi4h));
    l4230=sqrt((X41leader2-Xi4h)*(X41leader2-Xi4h)+(Y41leader2-Yi4h)*(Y41leader2-Yi4h));
    l413d=Dd(2)+D2+g;l423d=g;

    R5leader2=R51leader2;
    l5130=sqrt((Xi1-Xi5h)*(Xi1-Xi5h)+(Yi1-Yi5h)*(Yi1-Yi5h));
```

```
l5230=sqrt((X51leader2-Xi5h)*(X51leader2-Xi5h)+(Y51leader2-Yi5h)*(Y51leader2-Yi5h));
l513d=Dd(2)+D2+g;l523d=g;
sim('Step312_4151'); a=[a ' 4151'];
case 4152
R4leader2=R41leader2;
l4130=sqrt((Xi1-Xi4h)*(Xi1-Xi4h)+(Yi1-Yi4h)*(Yi1-Yi4h));
l4230=sqrt((X41leader2-Xi4h)*(X41leader2-Xi4h)+(Y41leader2-Yi4h)*(Y41leader2-Yi4h));
l413d=Dd(2)+D2+g;l423d=g;

R5leader2=R52leader2;
l5130=sqrt((Xi2-Xi5h)*(Xi2-Xi5h)+(Yi2-Yi5h)*(Yi2-Yi5h));
l5230=sqrt((X52leader2-Xi5h)*(X52leader2-Xi5h)+(Y52leader2-Yi5h)*(Y52leader2-Yi5h));
l513d=Dd(2)+D2+g;l523d=g;
sim('Step312_4152'); a=[a '4152'];
case 4153
R4leader2=R41leader2;
l4130=sqrt((Xi1-Xi4h)*(Xi1-Xi4h)+(Yi1-Yi4h)*(Yi1-Yi4h));
l4230=sqrt((X41leader2-Xi4h)*(X41leader2-Xi4h)+(Y41leader2-Yi4h)*(Y41leader2-Yi4h));
l413d=Dd(2)+D2+g;l423d=g;

R5leader2=R53leader2;
l5130=sqrt((Xi3-Xi5h)*(Xi3-Xi5h)+(Yi3-Yi5h)*(Yi3-Yi5h));
l5230=sqrt((X53leader2-Xi5h)*(X513eader2-Xi5h)+(Y53leader2-Yi5h)*(Y53leader2-Yi5h));
l513d=Dd(2)+D2+g;l523d=g;
sim('Step312_4153'); a=[a '4153'];
case 4154
R4leader2=R41leader2;
l4130=sqrt((Xi1-Xi4h)*(Xi1-Xi4h)+(Yi1-Yi4h)*(Yi1-Yi4h));
l4230=sqrt((X41leader2-Xi4h)*(X41leader2-Xi4h)+(Y41leader2-Yi4h)*(Y41leader2-Yi4h));
l413d=Dd(2)+D2+g;l423d=g;

R5leader2=R54leader2;
l5130=sqrt((Xi4-Xi5h)*(Xi4-Xi5h)+(Yi4-Yi5h)*(Yi4-Yi5h));
l5230=sqrt((X54leader2-Xi5h)*(X54leader2-Xi5h)+(Y54leader2-Yi5h)*(Y54leader2-Yi5h));
l513d=Dd(2)+D2+g;l523d=g;
sim('Step312_4154'); a=[a '4154'];
case 4251
R4leader2=R42leader2;
l4130=sqrt((Xi2-Xi4h)*(Xi2-Xi4h)+(Yi2-Yi4h)*(Yi2-Yi4h));
l4230=sqrt((X42leader2-Xi4h)*(X42leader2-Xi4h)+(Y42leader2-Yi4h)*(Y42leader2-Yi4h));
l413d=Dd(2)+D2+g;l423d=g;

R5leader2=R51leader2;
l5130=sqrt((Xi1-Xi5h)*(Xi1-Xi5h)+(Yi1-Yi5h)*(Yi1-Yi5h));
l5230=sqrt((X51leader2-Xi5h)*(X51leader2-Xi5h)+(Y51leader2-Yi5h)*(Y51leader2-Yi5h));
l513d=Dd(2)+D2+g;l523d=g;
sim('Step312_4251'); a=[a ' 4251'];
case 4252
R4leader2=R42leader2;
l4130=sqrt((Xi2-Xi4h)*(Xi2-Xi4h)+(Yi2-Yi4h)*(Yi2-Yi4h));
l4230=sqrt((X42leader2-Xi4h)*(X42leader2-Xi4h)+(Y42leader2-Yi4h)*(Y42leader2-Yi4h));
l413d=Dd(2)+D2+g;l423d=g;

R5leader2=R52leader2;
l5130=sqrt((Xi2-Xi5h)*(Xi2-Xi5h)+(Yi2-Yi5h)*(Yi2-Yi5h));
l5230=sqrt((X52leader2-Xi5h)*(X52leader2-Xi5h)+(Y52leader2-Yi5h)*(Y52leader2-Yi5h));
l513d=Dd(2)+D2+g;l523d=g;
```

```
 sim('Step312_4252'); a=[a ' 4252'];
case 4253
 R4leader2=R42leader2;
 l4130=sqrt((Xi2-Xi4h)*(Xi2-Xi4h)+(Yi2-Yi4h)*(Yi2-Yi4h));
 l4230=sqrt((X42leader2-Xi4h)*(X42leader2-Xi4h)+(Y42leader2-Yi4h)*(Y42leader2-Yi4h));
 l413d=Dd(2)+D2+g;l423d=g;

 R5leader2=R53leader2;
 l5130=sqrt((Xi3-Xi5h)*(Xi3-Xi5h)+(Yi3-Yi5h)*(Yi3-Yi5h));
 l5230=sqrt((X53leader2-Xi5h)*(X53leader2-Xi5h)+(Y53leader2-Yi5h)*(Y53leader2-Yi5h));
 l513d=Dd(2)+D2+g;l523d=g;
 sim('Step312_4253'); a=[a ' 4253'];
case 4253
 R4leader2=R42leader2;
 l4130=sqrt((Xi2-Xi4h)*(Xi2-Xi4h)+(Yi2-Yi4h)*(Yi2-Yi4h));
 l4230=sqrt((X42leader2-Xi4h)*(X42leader2-Xi4h)+(Y42leader2-Yi4h)*(Y42leader2-Yi4h));
 l413d=Dd(2)+D2+g;l423d=g;

 R5leader2=R54leader2;
 l5130=sqrt((Xi4-Xi5h)*(Xi4-Xi5h)+(Yi4-Yi5h)*(Yi4-Yi5h));
 l5230=sqrt((X54leader2-Xi5h)*(X54leader2-Xi5h)+(Y54leader2-Yi5h)*(Y54leader2-Yi5h));
 l513d=Dd(2)+D2+g;l523d=g;
 sim('Step312_4254'); a=[a ' 4254'];
case 4351
 R4leader2=R43leader2;
 l4130=sqrt((Xi3-Xi4h)*(Xi3-Xi4h)+(Yi3-Yi4h)*(Yi3-Yi4h));
 l4230=sqrt((X43leader2-Xi4h)*(X43leader2-Xi4h)+(Y43leader2-Yi4h)*(Y43leader2-Yi4h));
 l413d=Dd(2)+D2+g;l423d=g;

 R5leader2=R51leader2;
 l5130=sqrt((Xi1-Xi5h)*(Xi1-Xi5h)+(Yi1-Yi5h)*(Yi1-Yi5h));
 l5230=sqrt((X51leader2-Xi5h)*(X51leader2-Xi5h)+(Y51leader2-Yi5h)*(Y51leader2-Yi5h));
 l513d=Dd(2)+D2+g;l523d=g;
 sim('Step312_4351'); a=[a ' 4351'];
case 4352
 R4leader2=R43leader2;
 l4130=sqrt((Xi3-Xi4h)*(Xi3-Xi4h)+(Yi3-Yi4h)*(Yi3-Yi4h));
 l4230=sqrt((X43leader2-Xi4h)*(X43leader2-Xi4h)+(Y43leader2-Yi4h)*(Y43leader2-Yi4h));
 l413d=Dd(2)+D2+g;l423d=g;

 R5leader2=R52leader2;
 l5130=sqrt((Xi2-Xi5h)*(Xi2-Xi5h)+(Yi2-Yi5h)*(Yi2-Yi5h));
 l5230=sqrt((X52leader2-Xi5h)*(X52leader2-Xi5h)+(Y52leader2-Yi5h)*(Y52leader2-Yi5h));
 l513d=Dd(2)+D2+g;l523d=g;
 sim('Step312_4352'); a=[a ' 4352'];
case 4353
 R4leader2=R43leader2;
 l4130=sqrt((Xi3-Xi4h)*(Xi3-Xi4h)+(Yi3-Yi4h)*(Yi3-Yi4h));
 l4230=sqrt((X43leader2-Xi4h)*(X43leader2-Xi4h)+(Y43leader2-Yi4h)*(Y43leader2-Yi4h));
 l413d=Dd(2)+D2+g;l423d=g;

 R5leader2=R53leader2;
 l5130=sqrt((Xi3-Xi5h)*(Xi3-Xi5h)+(Yi3-Yi5h)*(Yi3-Yi5h));
 l5230=sqrt((X53leader2-Xi5h)*(X53leader2-Xi5h)+(Y53leader2-Yi5h)*(Y53leader2-Yi5h));
 l513d=Dd(2)+D2+g;l523d=g;
 sim('Step312_4353'); a=[a ' 4353'];
case 4351
```

```
    R4leader2=R43leader2;
    l4130=sqrt((Xi3-Xi4h)*(Xi3-Xi4h)+(Yi3-Yi4h)*(Yi3-Yi4h));
    l4230=sqrt((X43leader2-Xi4h)*(X43leader2-Xi4h)+(Y43leader2-Yi4h)*(Y43leader2-Yi4h));
    l413d=Dd(2)+D2+g;l423d=g;

    R5leader2=R54leader2;
    l5130=sqrt((Xi4-Xi5h)*(Xi4-Xi5h)+(Yi4-Yi5h)*(Yi4-Yi5h));
    l5230=sqrt((X54leader2-Xi5h)*(X54leader2-Xi5h)+(Y54leader2-Yi5h)*(Y54leader2-Yi5h));
    l513d=Dd(2)+D2+g;l523d=g;
    sim('Step312_4354'); a=[a ' 4354'];
   case 5154
    l5130=sqrt((Xi1-Xi5h)*(Xi1-Xi5h)+(Yi1-Yi5h)*(Yi1-Yi5h));
    l5230=sqrt((Xi4-Xi5h)*(Xi4-Xi5h)+(Yi4-Yi5h)*(Yi4-Yi5h));
    dis41=sqrt((Xi4-Xi1)*(Xi4-Xi1)+(Yi4-Yi1)*(Yi4-Yi1));
    if dis41 >= 2*(Dd(2)+D2)
       l513d=dis41-Dd(2)-D2+g;
    else
       l513d=Dd(2)+D2+g;
    end
    l523d=Dd(2)+D2+g;
    sim('Step312_5154'); a=[a ' 5154'];
   case 5254
    l5130=sqrt((Xi2-Xi5h)*(Xi2-Xi5h)+(Yi2-Yi5h)*(Yi2-Yi5h));
    l5230=sqrt((Xi4-Xi5h)*(Xi4-Xi5h)+(Yi4-Yi5h)*(Yi4-Yi5h));
    dis42=sqrt((Xi4-Xi2)*(Xi4-Xi2)+(Yi4-Yi2)*(Yi4-Yi2));
    if dis42 >= 2*(Dd(2)+D2)
       l513d=dis42-Dd(2)-D2+g;
    else
       l513d=Dd(2)+D2+g;
    end
    l523d=Dd(2)+D2+g;
    sim('Step312_5254'); a=[a ' 5254'];
   case 5354
    l5130=sqrt((Xi3-Xi5h)*(Xi3-Xi5h)+(Yi3-Yi5h)*(Yi3-Yi5h));
    l5230=sqrt((Xi4-Xi5h)*(Xi4-Xi5h)+(Yi4-Yi5h)*(Yi4-Yi5h));
    dis43=sqrt((Xi4-Xi3)*(Xi4-Xi3)+(Yi4-Yi3)*(Yi4-Yi3));
    if dis43 >= 2*(Dd(2)+D2)
       l513d=dis43-Dd(2)-D2+g;
    else
       l513d=Dd(2)+D2+g;
    end
    l523d=Dd(2)+D2+g;
    sim('Step312_5354'); a=[a ' 5354'];
end
 i=i+1;
 m=size(xyo1,1);
 tr(i)=xyo1(m,1);

 Xi1=xyo1(m,2); Yi1=xyo1(m,3); Oi1=xyo1(m,4);
 Xi2=xyo2(m,2); Yi2=xyo2(m,3); Oi2=xyo2(m,4);
 Xi3=xyo3(m,2); Yi3=xyo3(m,3); Oi3=xyo3(m,4);
 Xi4=xyo4(m,2); Yi4=xyo4(m,3); Oi4=xyo4(m,4);
 Xi5=xyo5(m,2); Yi5=xyo5(m,3); Oi5=xyo5(m,4);
 xyo1(:,1)=xyo1(:,1)+t;
 xyo2(:,1)=xyo2(:,1)+t;
 xyo3(:,1)=xyo3(:,1)+t;
 xyo4(:,1)=xyo4(:,1)+t;
 xyo5(:,1)=xyo5(:,1)+t;
```

```
    xyo1f=[xyo1f;  xyo1];
    xyo2f=[xyo2f;  xyo2];
    xyo3f=[xyo3f;  xyo3];
    xyo4f=[xyo4f;  xyo4];
    xyo5f=[xyo5f;  xyo5];
    t=t+tr(i); collision=0;
end
```

Appendix D

Organisation of the accompanying CD-ROM for video clips

This appendix presents the organisation of the accompanying CD-ROM for simulation and experimental video clips. Video clips are arranged in folders corresponding to the chapters of the thesis.

Folder *C3*: Simulations in Chapter 3

- *VRT31.avi*: Simulation 3.1 – Case of VRT control, collision happens at $t = 0.25s$
- *VHRT31.avi*: Simulation 3.1 – Case of VHRT control, no collision.
- *VRT32.avi*: Simulation 3.2 – Case of VRT control, impossible to obtain a desired formation
- *VHRT32.avi*: Simulation 3.2 – Case of VHRT control, possible to obtain a desired formation

Folder *C4*: Simulations in Chapter 4

- *formation41.avi*: Simulation 4.1 – Line formation
- *formation42.avi*: Simulation 4.2 – Case of singularities

Folder *C5*: Simulations in Chapter 5

- *formation51.avi*: Simulation 5.1 – Line Formation
- *formation52.avi*: Simulation 5.2 – Line Formation – No possibility of collisions
- *formation53.avi*: Simulation 5.3 – Diamond-like formation
- *formation54.avi*: Simulation 5.4 – Wedge formation

Folder *C6*: Simulations and experiment in Chapter 6

- *formation6_Cen.avi*: Simulation 6.1 – Centralised Control
- *formation6_Dec.avi*: Simulation 6.1 – Decentralised Control
- *formation_change.avi*: Simulation 6.2 – Three robots in various formations
- *Amigo2.avi*: Experiment – Two Amigo Mobile robots