Multi-Path Planning and Multi-Body Constrained Attitude Control

Innocent Okoloko

Dissertation presented for the degree of Doctor of Philosophy in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof Anton H. Basson

Co-supervisor: Dr Yoonsoo Kim

December 2012

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date:.....2012/12.....

Abstract

Multi-Path Planning and Multi-Body Constrained Attitude Control

I. Okoloko

Dissertation: PhD

December 2012

This research focuses on the development of new efficient algorithms for multi-path planning and multi-rigid body constrained attitude control. The work is motivated by current and future applications of these algorithms in: intelligent control of multiple autonomous aircraft and spacecraft systems; control of multiple mobile and industrial robot systems; control of intelligent highway vehicles and traffic; and air and sea traffic control.

We shall collectively refer to the class of mobile autonomous systems as “agents”. One of the challenges in developing and applying such algorithms is that of complexity resulting from the nontrivial agent dynamics as agents interact with other agents, and their environment. In this work, some of the current approaches are studied with the intent of exposing the complexity issues associated them, and new algorithms with reduced computational complexity are developed, which can cope with interaction constraints and yet maintain stability and efficiency.

To this end, this thesis contributes the following new developments to the field of multi-path planning and multi-body constrained attitude control:

- The introduction of a new LMI-based approach to collision avoidance in 2D and 3D spaces.

- The introduction of a consensus theory of quaternions by applying quaternions directly with the consensus protocol for the first time.

- A consensus and optimization based path planning algorithm for multiple autonomous vehicle systems navigating in 2D and 3D spaces.

- A proof of the consensus protocol as a dynamic system with a stochastic plant matrix.

- A consensus and optimization based algorithm for constrained attitude synchronization of multiple rigid bodies.

- A consensus and optimization based algorithm for collective motion on a sphere.

Uittreksel

Multipad-beplanning en Multi-liggaam Beperkte Standbeheer

("Multi-Path Planning and Multi-Body Constrained Attitude Control")

I. Okoloko

Proefskrif: PhD

Desember 2012

Hierdie navorsing fokus op die ontwikkeling van nuwe koste-effektiewe algoritmes, vir multipad-beplanning en veelvuldige starre-liggaam beperkte standbeheer. Die werk is gemotiveer deur huidige en toekomstige toepassing van hierdie algoritmes in: intelligente beheer van veelvuldige outonome vliegtuig- en ruimtevaartuigstelsels; beheer van veelvuldige mobiele en industrile robotstelsels; beheer van intelligente hoofwegvoertuie en verkeer; en in lug- en see-verkeersbeheer.

Ons sal hier "agente" gebruik om gesamentlik te verwys na die klas van mobiele outonome stelsels. Een van die uitdagings in die ontwikkeling en toepassing van sulke algoritmes is die kompleksiteit wat spruit uit die nie-triviale agentdinamika as gevolg van die interaksie tussen agente onderling, en tussen agente en hul omgewing. In hierdie werk word sommige huidige benaderings bestudeer met die doel om die kompleksiteitskwessies wat met hulle geassosieer word, bloot te lê. Verder word nuwe algoritmes met verminderde berekeningskompleksiteit ontwikkel. Hierdie algoritmes kan interaksie-beperkings hanteer, en tog stabiliteit en doeltreffendheid behou.

Vir hierdie doel dra die proefskrif die volgende nuwe ontwikkelings by tot die gebied van multipad-beplanning van multi-liggaam beperkte standbeheer:

- Die voorstel van 'n nuwe LMI-gebasseerde benadering tot botsingsvermyding in 2D en 3D ruimtes.
- Die voorstel van 'n konsensus-teorie van "quaternions" deur "quaternions" vir die eerste keer met die konsensusprotokol toe te pas.
- 'n Konsensus- en optimeringsgebaseerde padbeplanningsalgoritme vir veelvoudige outonome voertuigstelsels wat in 2D en 3D ruimtes navigeer.
- Die bewys van 'n konsensusprotokol as 'n dinamiese stelsel met 'n stochastiese aanlegmatriks.
- 'n Konsensus- en optimeringsgebaseerde algoritme vir beperkte stand sinchronisasie van veelvoudige starre liggame.

• ’n Konsensus- en optimeringsgebaseerde algoritme vir kollektiewe beweging op ’n sfeer.

Acknowledgements

I would like to express my thanks to my supervisors, for giving me this opportunity.

I gratefully thank the external and internal examiners for their very detailed comments, corrections and recommendations, which greatly helped to improve the thesis.

My appreciations to Professors Monday Ikhile and Samuel Ogbonmwan, both of University of Benin, Prof. Ben Sebitosi of Stellenbosch University, Prof. Hani Hagras of Essex University, and Mr. E. N. O. Adimora, for their support for my academic career.

My gratitudes to my lovely wife Helen, my family and my friends, especially those of the household of faith, for moral support.

Finally, my special thanks to the CSIR South Africa, and to the Harry Crossley Foundation, for the financial support for this work.

Dedications

...to thee the only true God, and Jesus Christ whom thou hast sent.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

AC	Attitude control
AHRS	Attitude and heading reference system
AI	Artificial intelligence
API	Application programming interface
CFTOC	Constrained finite time optimal controller
DHCP	Dynamic host configuration protocol
IP	Internet protocol
LMI	Linear matrix inequalities
MILP	Mixed integer linear programming
MPC	Model predictive control
PID	Proportional-integral-derivative control
Q-CAC	Quadratically constrained attitude control
RoboCup	The robot soccer league
SDP	Semidefinite programming
ssl-vision	RoboCup small-sized league vision system
UAV	Unmanned aerial vehicle
UDP	User datagram protocol
VTOL	Vertical take-off and landing
WiFi	Wireless fidelity
WLAN	Wireless local area network
2D	Two-dimensional Euclidean space
3D	Three-dimensional Euclidean space

Notation

a^{ij}, a_{ij}	Elements of the adjacency matrix of $\mathcal{G}$
α^i	Assigned role of vehicle i
$\tilde{A}$	Cone avoidance constraint matrix
$\mathcal{A}_{\mathcal{G}}$	Adjacency matrix of $\mathcal{G}$
$\tilde{\mathcal{A}}_{\mathcal{G}}$	Weighted adjacency matrix of $\mathcal{G}$

$\mathfrak{A}$	The set of roles for all vehicles
b^{ij}	Input bias of the consensus protocol
$\tilde{\mathcal{B}}_{\mathcal{G}}$	Incidence matrix of $\mathcal{G}$
$\mathfrak{B}$	Permutation matrix
$\mathcal{C}$	The consensus space for **x** or **q**, $\mathcal{C} = \{\mathbf{x} \mid x^1 = x^2 = \cdots = x^n\}$
$\mathfrak{S}$	System of agents
$\mathfrak{C}$	System of agents' constraints
$\mathfrak{C}_{\mathfrak{C}\mathfrak{S}}$	Constrained control problem of $\mathfrak{S}$
ς	Rule for updating α
D	Attitude control plant matrix, $\mathbf{D} \in \mathcal{S}^m$
D^{ij}	Euclidean distance between vehicles i and j
$\mathcal{D}_{\mathcal{G}}$	Degree matrix of $\mathcal{G}$
$\mathcal{E}^i$	Width of safety region of a vehicle i
$\mathcal{E}$	The set of edges of $\mathcal{G}$
$\sum$	Used in §1.3.1 as complete strategy
$\mathcal{F}^B_{SCi}$	Fixed body coordinate frame with origin at SC_i's centre
$\mathcal{F}^I_{SCi}$	SC_i's position with reference to inertial coordinates
g	Guard for updating α
$\mathcal{G}$	A graph
i	Label of a vehicle, robot or spacecraft
$\mathbf{I}_n$	The $n \times n$ identity matrix
1	A vector all of whose elements are 1
J	Cost of performing a task
L	Incremental cost of performing a task
L	Laplacian matrix
$\mathfrak{L}$	Laplacian-like stochastic matrix for unit vector control
$\mathcal{L}_{\mathcal{G}}$	Laplacian matrix of $\mathcal{G}$
$\mathcal{N}^i$	Neighborhood of vehicle i
0	A vector all of whose elements are zeros
P	Laplacian-like stochastic matrix for quaternion vector control
$\mathfrak{P}$	Perron matrix
q	Stacked vector of more than one quaternion vectors
q^i	Attitude quaternion vector of SC_i
$\bar{q}^i$	Vector part of q^i
q^i_4	Scalar part of q^i
$(q^{ij})^{off}$	Offset quaternion of SC_j from SC_i
$\mathbf{q}^{off}$	Stacked vector of offset quaternions
r	$r^* + \mathcal{E}$
r^*	Radius of **S**

$\mathbf{R}$	Rotation matrix
$\mathbf{R}_i$	Rotation matrix corresponding to q^i
$\mathbf{S}$	Bounding sphere or circle, of a vehicle or obstacle
SC	Spacecraft
SC_i	Spacecraft number i
$SE(3)$	The special Euclidean group
$\mathcal{S}^m$	The set of $m \times m$ positive definite matrices ($\mathcal{S}^m > 0$)
T	Horizon time for completing a task
τ	Control torque
u^i	Control input of vehicle i
$\mathbf{u}$	Stacked vector of control inputs of more than one vehicle
v^i	Attitude vector or direction of motion of vehicle i
v^i_{obs}	An obstacle vector emanating from vehicle i
v^{ij}_{obs}	Bounding obstacle vector of vehicle i emanating from j
$v^B_{obs_i}$	Vector of obstacle in $\mathcal{F}^B_{SCi}$
$v^I_{obs_i}$	Vector of obstacle in $\mathcal{F}^I_{SCi}$
$v^I_{obs_i \cdot j}$	Vector of the j^{th} obstacle in $\mathcal{F}^I_{SCi}$
$v^B_{cam_i}$	Vector of SC_i's camera in $\mathcal{F}^B_{SCi}$
$v^I_{cam_i}$	Vector of SC_i's camera in $\mathcal{F}^I_{SCi}$
V	Terminal cost of performing a task
$\mathcal{V}$	The set of vertices of $\mathcal{G}$
w	Angular velocity
x^i	Position vector of vehicle number i
$(x^{ij})^{off}$	Position offset vector of vehicles i and j
$\hat{x}$	Unit vector corresponding to vector x.
$\mathbf{x}$	Stacked vector of more than one position vectors
Δt	Discretization time step
Ω, Π	Quaternion dynamics plant matrix
$\mathbf{z}$	State vector of Markov chain
$>$	$A > 0$ indicates that A is a positive definite matrix
$\geq$	$A \geq 0$ indicates that A is a positive semidefinite matrix

Operators

$\otimes$	Kronecker multiplication
$\ominus$	Quaternion difference
$\in$	Is a member of a set
$\notin$	Not a member of a set
$\forall$	For all
$\ll$	Very much less than

Chapter 1

Introduction

1.1 Background and Motivation

Imagine the future, an automated city transportation system, with thousands of vehicles, sensors, actuators, processors, software, and people, working together to optimize the safety, security, and throughput of the system. Or, consider an emergency situation, such as crime, hostage, or disaster (e.g. earthquake, tsunami or hurricane). Several autonomous robots have been deployed to the scene, each equipped with sensors, communication equipment, and medical equipment. They are to cooperate in a mission (e.g. search and rescue), the mission must be completed efficiently in a specified time, and the search area must be completely covered. Again, consider a fully autonomous industrial plant (refinery, or other manufacturing plant), where there are thousands of sensors, actuators, processors and software, cooperating and coordinating their activities to perform a set of complex tasks, which when completed, the overall goal for the plant is achieved.

The three examples given above describe the current trends, and the near future of *multi-agent* systems. Multi-agent systems are autonomous systems consisting of two or more dynamic (or static) agents, which interact and cooperate, to achieve a set of objectives or goals, which are beyond the capability of a single agent. The agents may be heterogeneous, if they are mobile they may have different kinematics and dynamics, their sensors and actuators may be highly diverse, and they may have different communication protocols. But they must cooperate to achieve the desired task.

Coordination and control of these networks of agents has attracted a lot of research in recent times due to the variety of current and future potential applications, and different control strategies have been employed in control and coordination of such systems. Next, we discuss the common control architectures briefly.

1.1.1 Multi-agent Control Architectures

As illustrated in Figure 1.1, three basic architectures are generally employed in control design for a multi-agent team, *centralized* (e.g. Yanakiev and Kanellakopoulos, 1996; Swaroop and Hedrick, 1996), *decentralized* (e.g. Barret and Lafortune, 2000; Sandell *et al.*, 1978), and *distributed* (e.g. Stothert and McLeod, 1997; Yamaguchi *et al.*, 2001; Fregene, 2002). In the diagrams, P represents a plant which is to be controlled, A_i $(i = 1, \cdots, n)$ represents agent number i, which is a component of a multivariable plant P. P_{A_i} represents the decentralized autonomous plant for agent i, K is a centralized controller and K_i is the decentralized controller for agent i.

More detailed description of the architectures are briefly presented next.

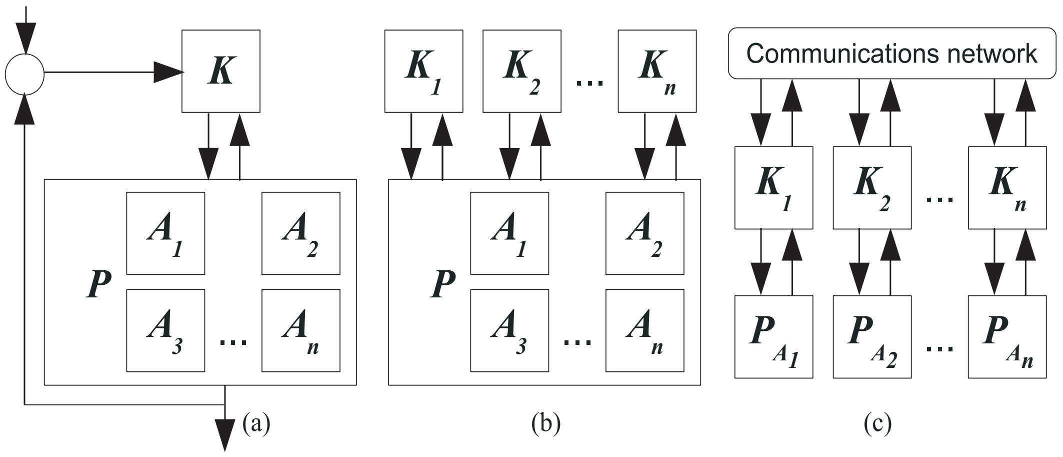


Figure 1.1: Multi-agent control architectures: (a) centralized, (b) decentralized, (c) distributed.

1.1.1.1 Centralized Architecture

In a centralized architecture, sometimes referred to in the control community as the "leader-follower" architecture, kinematics and dynamics of all agents A_i are modeled by using a single plant P. All planning, execution, control and monitoring of tasks are performed by a single controller K. P and K can be implemented on an agent in the group designated as a "leader" agent, or on an external supervisor computer. There are many systems for which this is the only option which can be implemented. For example, consider a team of remotely controlled vehicles which have no individual onboard processing and memory, but their actuators can be driven by external commands sent by a computer via wireless communication. The advantage of this approach is that of simplicity in high level design, and for this simplicity it is sometimes an attractive choice at first sight.

However, in certain circumstances, in the long run it becomes considerably more difficult to have a fully centralized system than to apply other approaches, because of the following associated demerits:

- For a very large number of agents, the complexities of P and K can become enormous. In such situations model reduction techniques may be applied (Davison, 1966; Aoki, 1968; Ramapriyan, 1970).

- The computational and information communication burden increases exponentially with the number of agents in the team, because only the centralized controller does the job of getting information from all agents, computing controls and communicating the controls to all the agents.

- The agents' dynamics and kinematics may be heterogeneous and so it may become extremely difficult to model the entire system using a singe centralized plant.

- The number of agents may be large and/or dynamically changing, so the optimal strategy may be hard to obtain and constantly updated (Lygeros and Godbole, 1994).

- In a basic centralized control system, the system is less robust because the degradation of the single main controller can result in an entire system breakdown.

1.1.1.2 Decentralized Architecture

In the decentralized architecture, a single plant P may be used to model the kinematics and dynamics of all agents A_i, however there is no single designated "leader" K, which directs and coordinates the actions of the other agents. Decision making and control is performed by each individual agent using its own controller K_i, based on the use of only local information. An advantage of using this scheme is that it requires minimal or no communication between agents, so calculations for the controller design may be simple. This can also become a disadvantage, as it may be less efficient compared to the global optimum which may be obtained by centralized control.

1.1.1.3 Distributed Architecture

An extremely decentralized architecture is the distributed architecture. This is a fully decentralized scheme in which each agent's dynamics and kinematics are modeled by its own plant P_{A_i}, with its corresponding controller K_i, while interagent communication is fully open. Because the agents must cooperate, decisions made by any K_i depends on sensed information, and also on information communicated to it over the network, concerning the overall goal G, and possibly the states of other agents. Suppose this scheme is fully realizable, then an ultimate dream of multi-agent control is achievable. The advantages are:

- It is naturally suitable for the design of hybrid heterogeneous large scale systems.
- It is suitable for systems in which the number of agents may be changing dynamically. Nodes (agents) may disappear and then reappear in the team randomly.
- It is flexible and fault tolerant, because the breakdown of a single agent in the team does not totally degrade the entire system. This is referred to as *graceful degradation*.
- It is naturally amenable to the current trend in pervasive and ubiquitous computing, because computers and sensors are becoming smaller, smarter, cheaper and network ready.

The disadvantages of using this scheme are:

- Communication complexity may increase exponentially with the number of agents in the team. Resolving these may require new methods.
- Computational complexity increases exponentially with the number of agents in the team. Resolving these may require new methods.
- If the system is absolutely decentralized and there is neither a leader nor communication among agents, then in certain situations it may become impossible to encode goals and coordinate the activities of agents to achieve these goals. This is possibly because the agents have no knowledge of the current goal, or how close they are to achieving the goal. Recall the refinery example given before. For such a situation, most of the agents may be static (non-mobile) agents. Assuming that is true, then the task of each agent can be implicitly encoded in the agent, and because it is precisely know a priori and it does not change, it may be simpler to implement a fully distributed scheme in such a situation. However, consider a team of unmanned aerial vehicles in combat situation and the goals are dynamically changing. A new goal is required for a new situation as time and circumstances dictate. Then the need for a supervisor or leader sometimes becomes imperative, even for the interim.

1.1.2 Practical Issues in Control System Design for Multi-agent Systems

Before designing control systems for multi-agent systems, it is important to decide which architecture is most suitable for the problem at hand. Note that controllers can be designed as *discrete* or *continuous* controllers, so there is the dual issue of control architecture, and type of controller.

The type of problem at hand usually suggests which combination of architecture and controller is more suitable to use. For example, Lygeros and Godbole (1994) apply a game theoretic approach to the hybrid multi-agent control problem. They identify that the solution to the game theoretic problem will produce a continuous control law. Pointing out the disadvantages of having a fully centralized or fully decentralized system, they suggest that *semi-autonomous* agent control is naturally suited for hybrid designs. Furthermore, they propose a hybrid controller that is arranged in a multi-level hierarchy where, at the lower level, each agent chooses its own optimal control strategy in the form of a continuous control law. At the higher level, discrete controllers are used in coordination and conflict resolution.

The controller is thus split into two layers, *supervisor* and *regulator*. The supervisor makes use of information to determine *strategy*, which is a sequence of control actions compatible with system capabilities. The regulator implements the strategy. Following this method, designing control architecture therefore involves decomposing the system into a subsystem hierarchy, specifying the subsystem interconnections, and determining the limits of the environmental inputs. Figure 1.2 depicts such a hybrid control scheme. Note that similar control schemes had been proposed before, for example by Sandell *et al.* (1978), Stothert and McLeod (1997), Barret and Lafortune (2000) and Fregene (2002), all of which are actually implementations of a *distributed-with-supervisor* approach. Because of the problems posed by the other architectures, this approach is an appealing choice to use for hybrid large scale multi-agent control design, for autonomous vehicle systems.

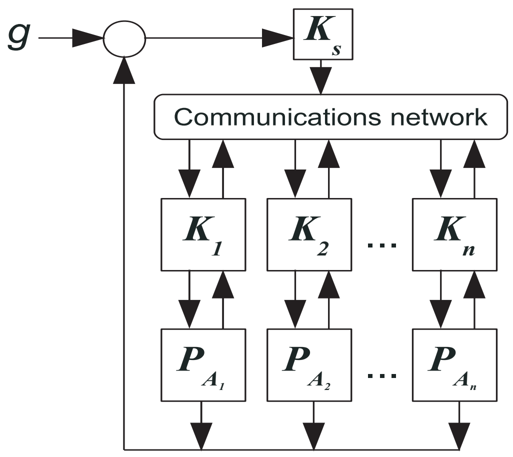


Figure 1.2: A distributed control scheme with a supervisor controller K_s.

Next, we consider some of the obvious bottlenecks which can hinder the successful implementation of such large scale hybrid networked multi-agent systems.

Following Jin (2007), we observe several impediments to successful practical implementation of large scale networked multi-agent systems. Even though designers make assumptions as to the capabilities of their multi-agent systems control designs, in most cases the simulations are usually done with a small number of agents, due to practical issues that require a closer consideration.

- However decoupled the agents are, their behaviours are coupled at some instance in time, because they must cooperate in order to accomplish a certain overall goal. The complexity resulting from this interaction topology among agents makes the dynamics of multi-agent systems more complicated than that of a single agent. The resulting nontrivial agent dynamics of each agent, as they interact, contribute more to the complexity. Although task decomposition and assignment are traditionally solved in a centralized manner, and some authors have addressed the issue of cooperative task allocation in a distributed fashion, how the complexity crisis can be resolved in distributed multi-agent systems is still an open research question.

- Networked multi-agent systems are based on telecommunications where, currently, data links between nodes are far from perfect. Random transmission delay, packet drops and uncertainty in connectivity topology all violate conventional assumptions in automatic control theory. How to realize effective control for multi-agent systems, despite these problems, is still an open research problem.

Two threads run through the arguments above. The first is that of system complexity. The reality of this problem is that computation required for multi-gent systems to operate in real time, becomes difficult for computers to undertake as the number of agents increases. However, the increase in the speed of computers offers some hope in tackling this problem. Secondly, distribution as a panacea to centralized computations in turn relies heavily on networked communication which has its own limitations.

Some researchers have tried to address these problems. For example, Minar *et al.* (1999), and Schoonderwoerd and Holland (1999), addressed the issue of multi-agent communication from the purely software agent paradigm. Olfati-Saber and Murray (2004) addressed consensus problems in networks of agents with switching topology and time delays. Also, Jin (2007) addressed the problem of communication in networked multi-agent system by proposing “multi-hop relay protocols”, which boost the process of consensus seeking and packet based state estimation over lossy communication networks, the results of which we believe can be practically applied to control of networked multi-agent systems. Moreover, recent applications of graph theoretic approaches to coordination of multi-agent systems provide a strong basis for tackling both the complexity and communication problems under a common framework. The problems of consensus, rendezvous, formation control and flocking, based on graph Laplacians, have been addressed by many authors, and the results are very promising. However, practical implementation of these algorithms in the presence of constraints is limited.

The arguments presented above motivate the intention for carrying out this work:

- To study some of the current approaches that have been employed for coordinated control of multi-agent systems, with the intent of understanding why the systems are complex, and how this complexity can limit their performance in implementation.
- Based on the above study, to develop new, computationally efficient algorithms, with improved performance for coordinated control of multi-agent systems.

1.2 Problem Formulation

Problem 1.1 (Graph theoretic and optimization based framework for characterizing a control problem of networked multi-agent systems, navigating under physical constraints). Given a system $\mathfrak{S}$, composed of a network of interacting agents on a mission to accomplish a desired overall goal, with agents interacting with non trivial dynamics and interaction topologies, subject to unavoidable constraints $\mathfrak{C}$, such as agent-to-agent interaction, find graph theoretic and optimization based control models for representing the constrained control problem $\mathfrak{C}_{\mathfrak{C}\mathfrak{S}}$ of $\mathfrak{S}$.

Problem 1.2 (Framework for solving the constrained control problem $\mathfrak{C}_{\mathfrak{C}\mathfrak{S}}$ 1.1.). Based on the model developed in Problem 1.1, develop control algorithms, with reduced computational complexity when compared to closely similar existing approaches (based on graph theory and optimization theory), as building blocks for solving the control problem of $\mathfrak{S}$.

The objective is to combine consensus and optimization theories in a new way to develop low cost algorithms that generate stable control laws, which can be proved theoretically, and verified at least by simulations, such that control of $\mathfrak{S}$ is stable and efficient.

1.3 Related Work

1.3.1 Notation for Coordinated Control of Multi-agent Systems

Although networked multi-agent systems are not restricted to mobile robots or autonomous vehicle systems, the development of control algorithms for networked multiple vehicle systems can be amenable to other aspects of networked multi-agent systems. The work in this thesis is concerned with multi-path planning and multi-body attitude control, therefore, autonomous multi-vehicle systems (or multiple mobile robot systems) will be used to describe the ideas presented.

In carrying out their tasks, we assume that each vehicle (or mobile robot) is able to communicate, at least partially, with other vehicles with which it has some communication link. This also includes vehicles within its sensor view or reach. From classical control theory, one way to define the dynamics of such a system is

$$\begin{aligned}\dot{x}^i =& f^i(x^i, u^i),\ x^i \in \mathbb{R}^n,\ u^i \in \mathbb{R}^m\ (i = 1 \cdots N),\\ y^i =& h^i(x^i),\ y^i \in SE(3),\end{aligned}$$

where $\dot{x}^i$, y^i, x^i are the state dynamics, measurement, and the discrete state respectively, of the i^{th} vehicle. The control input is u^i, f^i is the function representing its system dynamics, y^i is system output, and the set of rigid body configurations is the special Euclidean group $SE(3)$.

Apart from position, the i^{th} vehicle has a discrete state $\alpha^i \in \mathfrak{A}$, representing its assigned role, which encodes its actions, and the relationship of this current action, to the overall desired goal. $\mathfrak{A}$ is the set of roles, whose elements are determined by the coordinated control problem at hand. Because the role can change at any time, a *change of role* of vehicle i is defined as a function of its current state and role,

$$\alpha^{i^{'}}(t) = \varsigma(x^i(t), \alpha^i(t)),$$

where $\alpha^i(t)$ is the role of vehicle i at time t, $x^i(t)$ is its state, and the function ς is an update rule.

Some approaches to networked multi-agent control research rely very much on the concepts of algebraic graph theory (Biggs, 1974; Godsil and Royle, 2001), and matrix theory (Horn and Johnson, 1985). Based on graph theory, the set of agents or vehicles is viewed as a connected graph, with a structure representing the spatial relationship between vehicles, and a topology representing their communication flow. The set of possible communication channels are thus represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each vertex $i \in \mathcal{V}$ represents vehicle i (or a state of i), and each edge $(i, j) \in \mathcal{E}$ represents a communication link between vehicles i and j. The *neighbourhood* $\mathcal{N}^i(\mathcal{G})$ of vehicle i is the set of vehicles within the visibility or communication range of vehicle i and $|\mathcal{N}^i(\mathcal{G})|$ is the number of vehicles in $\mathcal{N}^i(\mathcal{G})$.

Given a collection of vehicles with states $x = (x^1, \cdots, x^N)$, and roles $\alpha = (\alpha^1, \cdots, \alpha^N)$, according to Murray (2006), tasks can be defined in terms of a cost function J, which in turn depends on two functions: the *incremental cost* L and *terminal cost* V, of performing the task. The cost is given as

$$J = \int L(x, \alpha, u)dt + V(x(T), \alpha(T)),$$

where T is the horizon time for completing the task. Furthermore, a *strategy* for a task is defined as an assignment of inputs u^i and roles to each of the vehicles, where the input to a vehicle is a function of the current state and required role. This assumes commands of the form

$$u^i = \gamma(x, \alpha),$$

where γ is a smooth function. To make a choice of roles, Klavins and Murray (2003) used the notion of a *guarded command language*, where a program is a set of commands of the form

$$\{g^i_j(x, \alpha) : \varsigma^i_j(x, \alpha)\},$$

where g^i_j is a guard that evaluates to true or false, and ς^i_j is a rule that defines how role α^i should be updated if g^i_j evaluates to true. The evolution of the role is defined by the following update law

$$\alpha^{i'} = \begin{cases} \varsigma^i(x^i, \alpha^i) & \text{if } g^i(x^i, \alpha^i) = \text{true} \\ \alpha^i & \text{otherwise,} \end{cases}$$

which should take place asynchronously for i vehicles. Alternatively it can be performed by a supervisor agent among the vehicles. Then,

$$strategy = control\ law + guarded\ commands.$$

Let $\sum^i$ be the overall strategy for vehicle i, then the *complete strategy* of the system is

$$\sum = \left(\sum^1, \cdots, \sum^N\right).$$

A control strategy is *centralized* if $\sum^i$ depends on the joint locations and roles of all vehicles, including non neighbours of vehicle i. Otherwise it is decentralized. Formally, a strategy is decentralized if

$$\begin{aligned} u^i(x, \alpha) =& u^i(x^i, \alpha^i, x^{-i}, \alpha^{-i}), \\ \{g^i_j(x, \alpha) : \varsigma^i_j(x, \alpha)\} =& \{g^i_j(x^i, \alpha^i, x^{-i}, \alpha^{-i}) : \varsigma^i_j(x^i, \alpha^i, x^{-i}, \alpha^{-i})\}, \end{aligned}$$

where α^{-i} are the roles and $x^{-i} = \{x^{j_1}, \cdots, x^{j_{m_i}} : j_k \in \mathcal{N}^i, m_i = |\mathcal{N}^i|\}$ are the states of vehicles in the neighbourhood of vehicle i.

1.3.2 Technology Overview

In this section, we survey some of the technologies that have been employed in coordinated control of multiple vehicles and mobile robots systems.

1.3.2.1 Spatio - Temporal Planning

Spatio-temporal planning (or path-planning), is the process where paths of the vehicles (or robots), and their positions with respect to time, are specified and controlled. Two main problems that can be solved by path planning are: (i) *rendezvous* which has been extensively studied (Ando *et al.*, 1999; Lin *et al.*, 2003; Ganguli *et al.*, 2005; Cortes *et al.*, 2006); and (ii) *coverage*.

Rendezvous is the process where all the vehicles or robots are required to arrive at a specified place at the same time, or within a specified time frame. Researchers have approached the problem by using *continuous time* and *discrete time* methodologies. For discrete time rendezvous, robots rendezvous according to discrete clock timing, either synchronously or asynchronously. Synchronous rendezvous drives each robot only at global clock ticks (Ando *et al.*, 1999; Cortes *et al.*, 2006; Ganguli *et al.*, 2005; Moreau, 2003). Example algorithms are the *circumcenter* algorithms (Barriere *et al.*, 2005; Flocchini *et al.*, 2005), where each robot moves towards the centre of the smallest circle containing itself and every robot it sees. In the asynchronous systems, robots converge without a global clock.

Continuous time rendezvous is based on a strategy called *cyclic pursuit* (Bruckstein *et al.*, 1995; Marshall *et al.*, 2004; Lin *et al.*, 2004; Smith *et al.*, 2005), where the robots are labelled $1, \cdots, n$, with each robot pursuing the next one, and the n^{th} robot pursuing the 1^{st}. Other approaches make use of *ellipsoidal cones* (Tiwari *et al.*, 2004; Bhattacharya *et al.*, 2004), and *polygon shortening flow* using a curve evolution method called *Euclidean curve shortening*.

Coverage is a process where the robots (or vehicles) are required to cooperatively distribute themselves to various positions on a specified area, in order to cover the area optimally. For example, consider a team of soccer playing robots, each player has a specific part of the pitch in which it should operate optimally. Coverage is also the idea used for optimal placement of distributed sensors, e.g. satellite clusters that must displace themselves to cover a particular area efficiently. See Choset (2001) for a survey on coverage algorithms.

1.3.2.2 Consensus and Cooperation

The consensus problem is that of driving the states of all vehicles (or robots) in a team, to a common value, by distributed protocols based on their communication network. Beginning with the work of Reynolds (1987), there has been a lot of interest in *flocking* and *swarming* behaviours of birds, fish and herds. Vicsek *et al.* (1995) proposed and analyzed a method for alignment of heading angles for multiple particles, using concepts from statistical mechanics. Motivated by that work, Jadbabaie *et al.* (2003) attempted to provide mathematical foundations for the emergent behaviours observed in (Vicsek *et al.*, 1995). They proposed a *discrete-time* consensus protocol based on stochastic matrix theory (Horn and Johnson, 1985). Given x^i, the state of vehicle i, the protocol is given as

$$x^i(k+1) = \sum_{x^j \in \mathcal{N}^i \cup \{i\}} a^{jk}(k) x^j(k), \tag{1.3.1}$$

where $\mathcal{N}^i$ is the set of vehicles in the neighbourhood of vehicle i at time k (i.e. other vehicles within the sensor view of i). With the assumption that $a^{ij}(k) \geq 0$, and $\sum a^{ij}(k) = 1$, vehicle i updates its state by using the weighted average of its heading and that of its neighbours. From the above, it is clear that the interaction topology of a network of vehicles can be conveniently represented using a graph.

Following the model given above, for systems modelled by first-order dynamics, the following continuous time first-order consensus protocol (or its variants) have been proposed (e.g., Olfati-Saber and Murray, 2004; Ren, 2004),

$$\dot{x}^i(t) = -\sum_{j \in \mathcal{N}(i)} b^{ij}(t)(x^i(t) - x^j(t)), \tag{1.3.2}$$

where $b^{ij}(t)$ is a positive weight. The system is said to have achieved consensus if $\lim_{t\to\infty} \|x^i(t) - x^j(t)\| = 0$ for $i \neq j$. The system is also said to have achieved average consensus if $x^i(t) \to \sum x^i(0)/n$ as $t \to \infty$, for any i, where n is the number of agents. Other protocols that depend on sensors with limited field of view, and state dependent graph Laplacians, have also been presented in the literature, but most of them are variants of (1.3.1) and (1.3.2). A mathematical derivation of the consensus algorithm (1.3.2) is developed in Section 2.3 as a part of this work.

For a time-invariant topology, it is sufficient that there exists a spanning tree: (i.e. the graph must be connected) for (1.3.1) and (1.3.2) to reach consensus (Ren and Beard, 2005; Moreau, 2005). While for average consensus, the topology must be at least *strongly connected* and *balanced*[1] (Lin *et al.*, 2005). If the topology changes with time (which is normal

[1]These terms will be explained in Chapter 2.

for wireless networked multi-agent systems), the above consensus protocols are still valid provided that at least one topology is strongly connected (Ren and Beard, 2005), or if there exists a spanning tree in each uniformly bounded time interval (Moreau, 2005).

An alternative iterative consensus algorithm that corresponds to the discrete-time version of system (1.3.2) is a Markov chain

$$\mathbf{z}(k+1) = \mathfrak{P}\mathbf{z}(k), \tag{1.3.3}$$

where $\mathfrak{P} = \mathbf{I} - \epsilon\mathbf{L}$ is a Perron matrix and the state transition probability matrix of the Markov chain, $\mathbf{I}$ is an identity matrix, $\epsilon \geq 0$ is a small step size, and $\mathbf{z}(k)$ is a column vector of the discrete states of the vehicles. For an arbitrary $\mathcal{G}$ with Laplacian $\mathbf{L}$, and a sufficiently small ϵ, $\mathfrak{P}$ satisfies the property $\sum_j p_{ij} = 1,\ p_{ij} \geq 0, \forall i, j, p_{ij} \in \mathfrak{P}$.

The effect of communication delays on consensus has also been studied by Lin *et al.* (2005), and Lawton and Beard (2002), where upper bounds of the delay margin were given for a fixed communication topology with uniform delay.

Consensus protocols have been a strong basis for formation control, rendezvous, distributed decision making in asynchronous peer-to-peer networks (Mehyar *et al.*, 2005), and robot synchronization (Rodriguez-Angeles and Nijmeijer, 2003). All of these applications rely on the assumption that the convergence speed of consensus seeking is fast enough. A few methods have been reported to improve the convergence speed, e.g. the work of Xiao *et al.* (2005) where convergence speed is improved by finding optimal weights associated with every communication link.

1.3.2.3 Formation Control

Formation is the process of having a team of autonomous vehicles assume and maintain a particular geometric pattern, or switch intermittently between different geometric patterns. It is applied in areas such as: (i) formation control of unmanned aircraft, spacecraft, and watercraft (Lewis and Tan, 1997; Balch and Arkin, 1998; Beard *et al.*, 2001; Fax, 2002; Kim *et al.*, 2003; Ren, 2004; Lin *et al.*, 2005); (ii) control of multiple robots (Lewis and Tan, 1997; Balch and Arkin, 1998; Feddema *et al.*, 2002); (iii) ground vehicle platooning

(Swaroop and Hedrick, 1996; Pant *et al.*, 2002); and (iv) satellite cluster positioning (Sabol *et al.*, 2001; Chichka, 2001; Yeh *et al.*, 2002).

Vehicle formations can either be represented as rigid structures by the use of gradient based control obtained from their structural potential functions, or by using vectors of relative positions of neighbouring vehicles with the aid of consensus-based controllers. Some fundamental results on formation control has been presented (Fax, 2002; Hu, 2003; Ren, 2004). Four main approaches are employed:

- *The virtual structure approach* (Lewis and Tan, 1997; Egerstedt *et al.*, 2001; Leonard and Fiorelli, 2001). In this approach, one "virtual leader" vehicle is synthesized based on all vehicles positions, which acts as the reference for the group, which is treated as a single rigid body. The position and trajectory of each vehicle is explicitly calculated relative to the position of the leader, and the formation configuration. While it is easy to describe the whole group and maintain accurate formation, this approach is only practical for small groups. This is because centralized data collection and processing are needed, with all its attendant problems of heavy information burden, as well as network and computational complexity. Distributed implementations of this approach are reported by Beard *et al.* (2001) and Olfati-Saber (2006).

- *Leader follower approach*, where an actual vehicle in the group acts as a designated leader for the other vehicles. Any error of the leader is propagated down the line of agents, a process known as "string stability". Obviously, this dependence on the team leader spells disaster for the team if the leader fails.

- *Distributed approach*, where there is no designated string "leader" for the team, and each vehicle solves its own assigned problem individually based on communication with other team members, as in the work of Fax (2002).

- *Behavior-based approach*, which is commonly used in the robotics community. The control law of each vehicle or robot is defined by a combination of predefined control actions corresponding to all possible robot states. As is always the case with behaviour based approaches to control, the controllers are flexible, but difficult to analyze quantitatively. Examples for this approach are reported by Balch and Arkin (1998), and Bauso *et al.* (2003).

1.3.2.4 Control over Communication Networks

On-board sensing and wireless communication links made wireless networked multi-agent systems a possibility. As observed earlier, today's communication networks are imperfect, due to transmission delays, packet drops, and dynamically changing topologies[2]. The problems worsen as the number of agents increases. In this section, some of the impacts of this problem on distributed control are reviewed. Figure 1.3 shows a typical networked multi-agent control system.

Traditional control theory makes the following assumptions, which cannot be satisfied by networked communication for the reasons given:

- Topology is assumed to be static, but the interaction topology of distributed mobile multi-agent systems is time-varying and highly dynamic.
- Connections are assumed to be reliable, but sampled signals are transmitted in data packets that may not be transmitted correctly. Connections exhibit unpredictable transmission delays and random packet drops.
- Bandwidth is assumed to be infinite, but communication channels can only transmit the data with certain precision under the constraint of limited bandwidth. Quantization and distortion must therefore be considered for system design and analysis.

These standard problems are constantly being addressed in networked control systems research, by combining tools from computer science, information theory, and communications engineering (Wong and Brockett, 1997; Altman *et al.*, 1999; Mitter, 2000; Feng, 2001; Walsh *et al.*, 2002). Brockett and Liberzon (2000) presented some significant results for estimation and stabilization of closed-loop systems over a communication channel with

[2]As with mobile agents.

finite bandwidth. Also, Elia and Mitter (2001), and Tatikonda and Mitter (2004), presented quantizers and optimal quantization that can help to improve on the number of bits reliably transmitted. Sinopoli *et al.* (2004) proposed different time-variant coding schemes for noisy and noiseless channels, while Paganini *et al.* (2001), and Liu and Goldsmith (2004), studied Kalman filtering of signals with intermittent observations. In addition, Liu and Goldsmith (2004) derived stochastic equations for the error covariance based on the Bernoulli packet dropping model. Furthermore, they formally showed a phase transition phenomenon for the expected value of error covariance with respect to the packet-dropping rate.

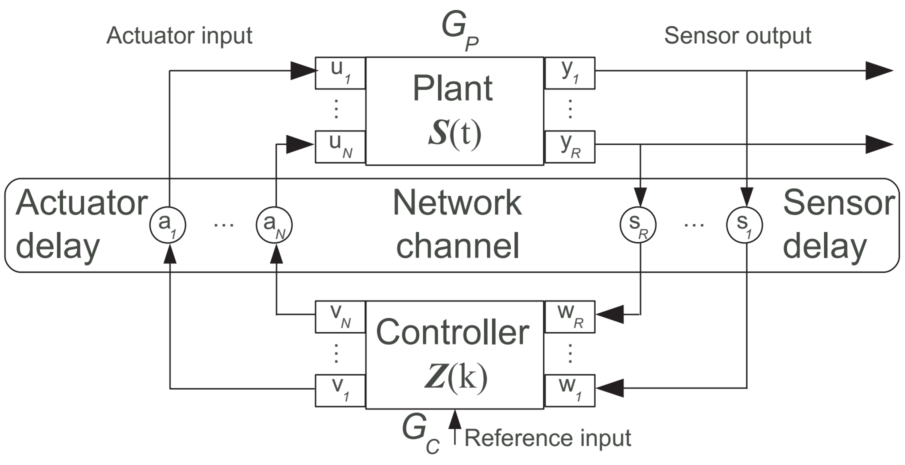


Figure 1.3: Closed-loop block diagram of networked control system with centralized control.

1.4 Thesis Outline

This section provides an overview of the contents of each subsequent chapter in this thesis.

In Chapter 2, the basic mathematical tools used in this thesis are presented. In addition, a derivation of the consensus theory, which is a core foundation of this research, is developed. It is formally shown that the consensus protocol is a dynamic system with a stochastic plant matrix, which acts as an attractor. The knowledge developed in this chapter is applied in the development of the consensus and optimization based multi-path planning algorithm presented in Chapter 3. The concepts of quadratically constrained attitude control (Q-CAC), semidefinite programming (SDP) and linear matrix inequalities (LMI) are also described in Chapter 3. The concepts developed in Chapter 3 lay the foundation for the consensus and optimization based constrained attitude control algorithm developed in Chapter 4. The ideas described in Chapter 3 and Chapter 4 come to play in the development of the multi-path planning algorithm for collective motion on a sphere, which is described in Chapter 5. Conclusions and future work are presented in Chapter 6 where the contributions are also listed. A practical implementation of the multi-path planning algorithm is presented in Appendix A.

Chapter 2

Mathematical Tools

In this chapter some of the mathematical tools used in this thesis, are presented. The ideas are mainly from graph theory and matrix theory. Section 2.1 introduces the basic ideas of graph theory and algebraic graph theory, which relates to this work, and is central in this review. Section 2.2 contains some results from the theory of nonnegative matrices, while section 2.3 presents a mathematical derivation of the consensus protocol, which is a core foundation of this research. It is formally proved that the consensus protocol in (1.3.2) is a dynamic system with a stochastic plant matrix.

2.1 Graph Theory

The navigation algorithms developed in this thesis are based on a graph theoretic approach, centered on the communication graph that results in a Laplacian matrix. The purpose of this section is to present a basic review of graph theory required to define the Laplacian matrix and introduce the theory of nonnegative matrices presented in Section 2.2.

2.1.1 Basic Graph Theory

A *graph* $\mathcal{G}$ is a mathematical structure $(\mathcal{V}, \mathcal{E})$ that consists of two finite sets: a set $\mathcal{V} = \{1, 2, \cdots, n\}$ of points called *vertices* or *nodes*; and a set $\mathcal{E} \subseteq \{(v_i, v_j) : v_i, v_j \in \mathcal{V}, j \neq i\}$ of connecting lines, called *edges* or *endpoints* of the vertices. More formally, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that there exist a function $f : \mathcal{E} \mapsto \mathcal{V} \times \mathcal{V}$. Graph $\mathcal{G}$ is said to be *undirected* if each edge is an unordered pair. It is otherwise called a *digraph* (directed graph).

Vertices are usually denoted either by the letters $u, v, \cdots$, or $v_1, v_2, \cdots, v_n$, or by numbers $1, 2, \cdots, n$, where n is the number of vertices. Edges $\mathcal{E}$ are usually denoted by the letters $e_1, e_2, \cdots, e_m$, where m is the number of edges. Edges may also be denoted by their endpoints (v_i, v_j) or $e(v_i, v_j)$, e.g. for the graph of Figure 2.1, $\mathcal{V} = \{1, 2, 3, 4, 5, 6\}$ and $\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$, where $e_1 = (1, 2)$ and $e_6 = (4, 5)$. The order of the arrangement of v_i and v_j in (v_i, v_j) matters for a digraph, but does not matter for an undirected graph.

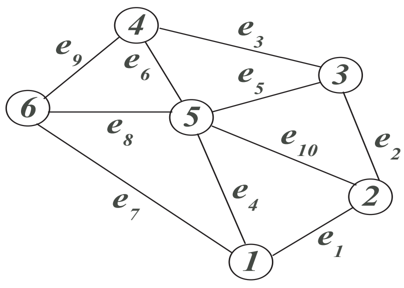


Figure 2.1: An undirected graph $\mathcal{G}$ with six vertices and ten edges.

The number of vertices $|\mathcal{V}|$, is called the *order*, and the number of edges $|\mathcal{E}|$, the *size*, respectively, of $\mathcal{G}$. The set of neighbours of node v_i, denoted by $\mathcal{N}_i = \{v_j \in \mathcal{V} : (v_i, v_j) \in \mathcal{E}\}$, is the set of nodes that have an edge directly connected to v_i. A *cluster* is a subset $J \subseteq \mathcal{V}$ of the nodes or vertices of the graph, and the set of neighbours of the cluster, $\mathcal{N}_J$ is defined by

$$\mathcal{N}_J := \bigcup_{V_i \in J} \mathcal{N}_i = \{v_j \in \mathcal{V} : v_i \in J, (v_i, v_j) \in \mathcal{E}\}.$$

The *null graph* has no edges and vertices. The *vertex-graph* is an edgeless graph having one vertex, e.g. vertex 3 in Figure 2.2 taken alone. The *loop-graph* consist of a single loop with its one vertex, e.g. vertex 4 in Figure 2.2, taken alone with its attaching loop edge e_4. The *link-graph* consists of a single link with two vertices, e.g. vertices 1 and 4, and the edge connecting them in Figure 2.2, taken alone. An *n-clique* is a loopless graph with exactly n vertices and $\frac{1}{2}n(n-1)$ edges, where n is a positive integer.

An edge (v_i, v_j) is said to be *incident* with vertices v_i and v_j. The *degree* or *valence* of a vertex is the number of edges incident with the vertex v, e.g. the degree of vertex 5 in Figure 2.1 is 5. Two vertices v_i and v_j are adjacent in $\mathcal{G}$ if they are directly connected by some edge e in $\mathcal{G}$, e.g. vertices 1 and 2 are adjacent in the graph in Figure 2.1.

The combination of the previous paragraphs leads us to introduce four basic types of graphs, based on some properties of graphs:

1. **General graphs**, can contain loops and parallel edges[1].

2. **Simple graphs**, may not have loops or parallel edges.

3. **Digraphs**, have no restrictions.

4. **Simple digraphs**, have no loops and no parallel edges.

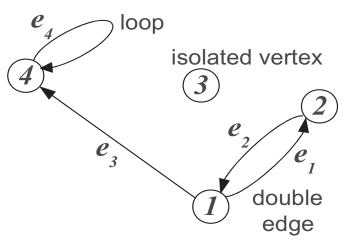


Figure 2.2: Special graph with an isolated vertex, loop and double edge.

The *in-degree* of a vertex of a digraph is the number of directed edges incident into that vertex v, and is defined as $deg_{in}(v_i) = \left\{\sum_{j=1}^{n} \left\|\tilde{b}_{ij}\right\| \middle| \tilde{b}_{ij} < 0\right\}$, where $\tilde{\mathcal{B}} = [\tilde{b}_{ij}]$ is the *incidence matrix* (defined in the next section). E.g. the in-degree of vertex 3 in Figure 2.3 (a) is 2, in (b) it is 3. The *out-degree* of a vertex of a digraph is the number of edges incident with the vertex v, but directed out of the vertex to other vertices, or back to itself in a loop.

[1]Multiple edges that are incident to two distinct vertices.

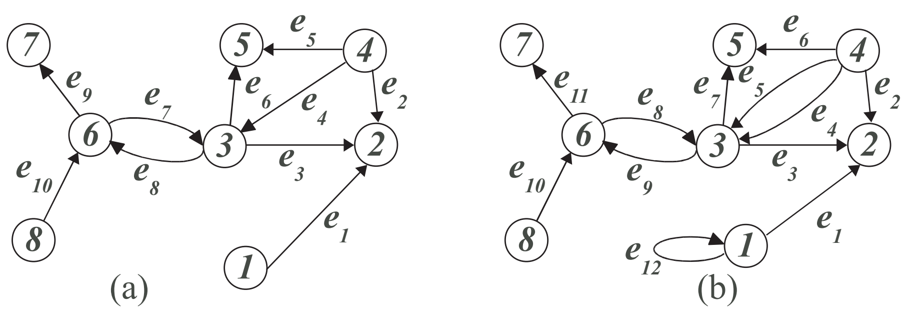


Figure 2.3: Digraphs (a) simple digraph (b) non simple digraph.

It is denoted by $deg_{out}(v_i) = \left\{ \sum_{j=1}^{n} \tilde{b}_{ij} | \tilde{b}_{ij} > 0 \right\}$, e.g. the out-degree of vertex 4 in Figure 2.3 (a) is 3.

Graph $\mathcal{G}$ is said to be *balanced* if and only if $deg_{in}(v_i) = deg_{out}(v_i)$ for each $v_i \in \mathcal{G}$. Consequently any undirected graph is balanced, the same does not automatically hold for digraphs. The number of neighbours of v_i is denoted by $|\mathcal{N}_i| = deg_{out}(v_i) + deg_{in}(v_i)$.

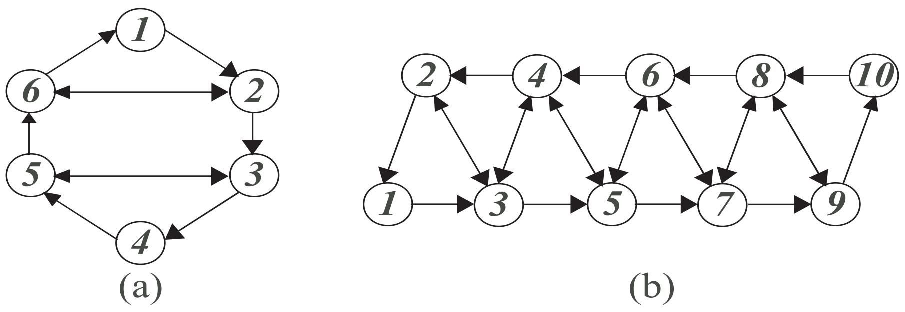


Figure 2.4: Examples of balanced digraphs.

A *walk* is said to exist from v_i to v_k in $\mathcal{G}$ if one can walk from v_i to v_k along some edge(s), with no restrictions. Thus, the walk is of the form $(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)$. A walk becomes a *trail* if there is a restriction that each $e \in \mathcal{E}$ be traversed at most once. A trail becomes a *path* if there is the further restriction that each $v \in \mathcal{V}$ be visited at most once. Furthermore, a *strong path* in a digraph is defined as a sequence of distinct vertices $[v_0, \cdots, v_r]$, where $(v_{i-1}, v_i) \in \mathcal{E}$ for any $i \in [1, \cdots, r]$, and r is the length of the path. If the above conditions are relaxed by having either (v_{i-1}, v_i) or $(v_i, v_{i-1}) \in \mathcal{E}$, then the path is called a *weak path*. And a *closed path* is a path which ends at the vertex it started from, while a *cycle* is a closed path that has at least three edges. A *Hamiltonian cycle* is a cycle that contains all the vertices of $\mathcal{G}$.

A digraph $\mathcal{G}$ is said to be *strongly connected* $(\mathcal{G}_{SC})$, if and only if any two distinct vertices can be connected via a strong path. Consequently, $\mathcal{G}$ is *weakly connected* $(\mathcal{G}_{WC})$ if any two distinct vertices can be connected via a weak path, otherwise, $\mathcal{G}$ is *disconnected* $(\mathcal{G}_{NC})$. If $\mathcal{G}$ is strongly connected and symmetric, then $\mathcal{G}$ is called *connected and symmetric* $(\mathcal{G}_{C/S})$. Also, $\mathcal{G}$ can be *disconnected and symmetric* $(\mathcal{G}_{NC/S})$, or *disconnected and non-symmetric* $(\mathcal{G}_{NC/NS})$. Figure 2.5 illustrates the connectivity and symmetry properties.

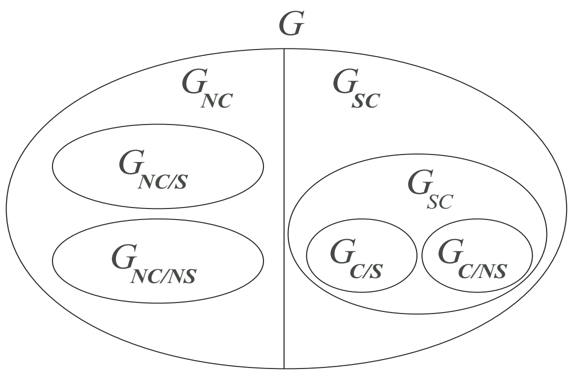


Figure 2.5: Connectivity and symmetry properties of a graph.

A digraph $\mathcal{G}$ that has no cycles is called *acyclic*, in this case there exists at least one $v_i \in \mathcal{V}$ such that $deg_{out}(v_i) = 0$. Consequently, a *tree* is a digraph that has no cycles. A *rooted directed spanning tree* of a digraph $\mathcal{G}$, is a subgraph $\mathcal{G}_r = (\mathcal{V}, \mathcal{E}_r)$, where $\mathcal{E}_r \subset \mathcal{E}$ connects all vertices in $\mathcal{G}$ such that each vertex in $\mathcal{G}_r$ has one and only one outgoing edge, except the root vertex. Alternatively, a rooted directed spanning tree of $\mathcal{G}$ can also be defined as $\mathcal{G}_r = (\mathcal{V}, \mathcal{E}_r)$, where $\mathcal{E}_r \subset \mathcal{E}$ connects all vertices in $\mathcal{G}$, such that each vertex in $\mathcal{G}_r$ has one and only one incoming edge, except the root vertex. An *induced maximal subgraph* is a subgraph of $\mathcal{G}$ that is not contained in another subgraph of $\mathcal{G}$. Finally, a *strong component* of $\mathcal{G}$ is an induced maximal subgraph that is strongly connected.

2.1.2 Basic Algebraic Graph Theory

To enable mathematical analysis and computation on graphs, they are usually represented in data structures. The transformation from diagrammatic representation results in various types of matrices or lists, which makes mathematical analysis directly amenable. Algebraic graph theory (Biggs, 1974; Godsil and Royle, 2001) is the study of graphs in relation to linear algebra, and some important matrix representations of graphs based on this theory are given below.

The *adjacency matrix* $\mathcal{A}_\mathcal{G} = [a_{ij}]$ of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of order n, is an $n \times n$ matrix with elements

$$a_{ij} = \begin{cases} 1 & \text{if } e(i,j) \in \mathcal{E} \\ 0 & \text{otherwise.} \end{cases}$$

For an undirected graph $\mathcal{A}_\mathcal{G}$ is symmetric due to the bidirectional property of edges. However, $\mathcal{A}_\mathcal{G}$ of a digraph $\mathcal{G}$ is symmetric if and only if $\mathcal{G}$ is symmetric. For example $\mathcal{A}_\mathcal{G}$ of the graph in Figure 2.1 is

$$\mathcal{A}_\mathcal{G} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}. \tag{2.1.1}$$

Observe in (2.1.1) that $\mathcal{A}_\mathcal{G} = \mathcal{A}_\mathcal{G}^T$. However, $\mathcal{A}_\mathcal{G}$ of the digraph in Figure 2.3 (a) is

$$\mathcal{A}_{\mathcal{G}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix},$$

which is obviously not symmetric.

The *out-degree matrix* $\mathcal{D}_{\mathcal{G}} = [d_{ik}]$ of a graph $\mathcal{G}$ of order n, is an $n \times n$ diagonal matrix with elements

$$d_{ik} = \begin{cases} \sum_j a_{ij} & \text{if } i = k \\ 0 & \text{if } i \neq k. \end{cases}$$

This is simply the diagonal matrix, with each diagonal element equal to the out-degree of the corresponding vertex. We can similarly define the *in-degree matrix* of $\mathcal{G}$. If the out-degree equals the in-degree for each vertex in $\mathcal{G}$, then $\mathcal{G}$ is symmetric, and $\mathcal{D}_{\mathcal{G}}$ is called the *degree matrix* of $\mathcal{G}$.

The *weighted adjacency matrix* $\tilde{\mathcal{A}}_{\mathcal{G}} = [\tilde{a}_{ij}]$ of a graph $\mathcal{G}$ of order n, is an $n \times n$ matrix with elements

$$\tilde{a}_{ij} = \begin{cases} w_{ij} & \text{if } e(i,j) \in \mathcal{E} \\ 0 & \text{otherwise,} \end{cases}$$

where w_{ij} is a positive weight associated with the edge (v_i, v_j). This weighting redefines the out-degree of v_i as the sum of the weights of edges incident from v_i, and the in-degree as the sum of the weights of edges incident into v_j. In the former definition of the adjacency matrix, w_{ij} is taken as 1.

The *incidence matrix* $\tilde{\mathcal{B}}_{\mathcal{G}} = [\tilde{b}_{ij}]$ of a graph $\mathcal{G}$ of order n, is an $n \times m$ matrix with elements

$$\tilde{b}_{ij} = \begin{cases} -1 & \text{if } e_j \text{ enters vertex } i \\ 1 & \text{if } e_j \text{ leaves vertex } i \\ 0 & \text{otherwise,} \end{cases}$$

where n is the number of vertices and m is the number of edges of $\mathcal{G}$, respectively. Matrix $\tilde{\mathcal{B}}$ is also called the *connectivity matrix* or *topology matrix* of $\mathcal{G}$.

The ***Laplacian*** *matrix* $\mathcal{L}_\mathcal{G} = [l_{ij}]$ of digraph $\mathcal{G}$ of order n, is the $n \times n$ matrix

$$\mathcal{L}_\mathcal{G} = \mathcal{D}_\mathcal{G} - \mathcal{A}_\mathcal{G}.$$

The *normalized adjacency matrix* is obtained by normalizing every row of $\mathcal{A}$ by the corresponding out-degree, and is defined as

$$\bar{\mathcal{A}}_\mathcal{G} = \mathcal{D}_\mathcal{G}^{-1}\mathcal{A}_\mathcal{G}.$$

To avoid division by zero, we set $d_{ii}^{-1} = 0$ if $deg_{out}(v_i) = 0$.

The *normalized Laplacian matrix* is defined as

$$\bar{\mathcal{L}}_\mathcal{G} = \mathcal{D}_\mathcal{G}^{-1}\mathcal{L}_\mathcal{G}.$$

For the digraph $\mathcal{G}$ of Figure 2.3 (a)

$$\mathcal{D}_\mathcal{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \mathcal{L}_\mathcal{G} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 3 & 0 & -1 & -1 & 0 & 0 \\ 0 & -1 & -1 & 3 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix},$$

$$\bar{\mathcal{A}}_{\mathcal{G}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 0 & 1/3 & 1/3 & 0 & 0 \\ 0 & 1/3 & 1/3 & 0 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix},$$

$$\bar{\mathcal{L}}_{\mathcal{G}} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1/3 & 1 & 0 & -1/3 & -1/3 & 0 & 0 \\ 0 & -1/3 & -1/3 & 1 & -1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1/2 & 0 & 0 & 1 & -1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}.$$

The next section describes the relationship between the connectivity of a graph, its adjacency matrix, and its Laplacian. This description is based on previous results (Horn and Johnson, 1985; Godsil and Royle, 2001) about the spectrum and rank of matrices $\mathcal{A}$ and $\mathcal{L}$.

2.2 Introduction to the Theory of Nonnegative Matrices

In the development of navigation algorithms, which form a part of dynamic systems, it is often required to mathematically analyse and prove the stability and convergence of an algorithm as a basic prerequisite for its feasibility. The stability of a dynamic system has a directly relationship with the nonnegativity of its plant matrix. This section gives a basic understanding of the relationships between the concepts of *irreducibility*, *connectivity* of a graph, and *nonnegativity* of a matrix.

The study of *nonnegative* matrices plays an important role in the analysis of dynamic systems, and three important types are *irreducible*, *stochastic* and *primitive* (or ergodic) matrices.

A *permutation matrix* $\mathfrak{B}$ is an elementary matrix which results by switching rows of the identity matrix, and $\mathfrak{B}$ is orthogonal, i.e. $\mathfrak{B}^T = \mathfrak{B}^{-1}$.

A matrix $\mathbf{A}$ is said to be *reducible* if there exists a permutation matrix $\mathfrak{B}$ such that

$$\mathfrak{B}^T \mathbf{A} \mathfrak{B} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix},$$

where $\mathbf{A}_{11}$, $\mathbf{A}_{21}$ and $\mathbf{A}_{22}$ are square matrices, and $\mathbf{0}$ is a square matrix of zero elements. The matrix is said to be *irreducible* if such a $\mathfrak{B}$ does not exist.

A *primitive matrix* $\mathbf{A}$ is a nonnegative matrix for which $\mathbf{A}^m$ is also positive[2] for some positive and sufficiently large m. A primitive matrix is irreducible and has only one eigenvalue of maximum modulus.

Two directed graphs $\mathcal{G}_x$ and $\mathcal{G}_y$ are said to be *isomorphic* if there exists a permutation matrix $\mathfrak{B}$ such that

$$\mathcal{A}_{\mathcal{G}_y} = \mathfrak{B}^T \mathcal{A}_{\mathcal{G}_x} \mathfrak{B}.$$

A *stochastic matrix* is a nonnegative square matrix whose rows or columns each consists of nonnegative real numbers, with each row or column summing to 1. Some matrices which do not strictly satisfy this definition are also regarded as stochastic, e.g. a Laplacian matrix.

The importance of this study lies on the realisation that the sensing topology of a team of agents may result in a weakly connected or disconnected graph, whose structure represents a communication topology wherein it is difficult, or impossible, to achieve consensus. Indeed the graph may contain subgraphs which are strong components (subgraphs whose vertices communicate). In such a graph, a vertex which is not at the head of any directed edge is said to be *initial*, and a vertex which is not at the tail of any directed edge is said to be *final*. If $\mathcal{G}$ is disconnected or weakly connected, $\mathcal{A}_{\mathcal{G}}$ is a reducible matrix.

[2] A matrix is positive or nonnegative if none of its eigenvalues is negative.

The next corollary (Godsil and Royle, 2001) establishes a relationship between the adjacency matrix of a graph, its connectivity, nonnegativity, and the irreducibility of its underlying digraph.

Corollary 2.1 The adjacency matrix of a graph $\mathcal{A}_{\mathcal{G}}$ (or any square matrix), is irreducible if its underlying digraph $\mathcal{G}$ is strongly connected (p. 175 in Godsil and Royle (2001)), otherwise it is reducible. Therefore for a strongly connected $\mathcal{G}$ $\mathcal{A}_{\mathcal{G}}$ is a nonnegative matrix.

Theorem 2.1, presented next, establishes a formal relationship between any nonnegative $n \times n$ matrix $\mathbf{A}$, its connectivity and its irreducibility.

Theorem 2.1. *Given a nonnegative $n \times n$ matrix $\mathbf{A}$, the following are equivalent* (Fax, 2002):

1. *$\mathbf{A}$ is irreducible*
2. *$\mathbf{A}^T$ is irreducible*
3. *$\mathcal{G}_{\mathbf{A}}$ is strongly connected*
4. *$(I_n + \mathbf{A})^{n-1}$ is positive definite*

From classical control theory, when the plant matrix of a dynamic system is nonnegative, the system is stable in the least. A conclusion can be drawn from Corollary 2.1 and Theorem 2.1 that as long as the communication graph of a team of communicating vehicles is strongly connected, convergence (consensus) can be achieved. If the underlying adjacency matrix $\mathbf{A}$ is reducible, a permutation matrix may be applied to realize a communication topology wherein it may be easier to achieve convergence.

2.3 Algebraic Derivation of the Consensus Protocol

In the development of consensus based navigation algorithms, it is often required to mathematically analyse and prove the stability or convergence of an algorithm. Most approaches to the proof of stability is to determine the eigenvalues of the resulting consensus based plant matrix. If the matrix is nonnegative, then the stability of the algorithm is certain.

This section presents a mathematical derivation of the consensus algorithm, protocol (1.3.2), using tools from algebra and geometry, purposely to give a clearer mathematical insight into the consensus algorithm; this insight can serve as an important ingredient for better understanding and further development of the consensus algorithm. The main aim is to provide another proof of the convergence of the consensus algorithm.

The first derivation in this section proves that circulant matrices lead to swirling motion which converges to a point. To show this, a circulant Perron matrix[3] was extracted from the flow of points on a set of spiralling lines (Theorem 2.2). This essentially proves that a circulant Laplacian matrix is convergent.

The second derivation shows by induction that a fully connected Laplacian matrix is also convergent. The known property of nonnegativity of the Laplacian matrix is established in the derivation. A set of rules or properties were obtained in the derivation which if a matrix satisfies, its stability is certain. These rules were later used in proving the convergence of the consensus based multipath planning algorithm which was developed in Chapter 3, in a new way. The derivations are presented next.

Consider two points P^1 and P^2 on a line. Suppose one wants to find a point P^q on the line segment between P^1 and P^2. Note that for clarity, superscript notations have been used here to identify points in order to allow the use of subscripts to denote the elements of a point. Therefore the superscript should not be confused with exponentiation.

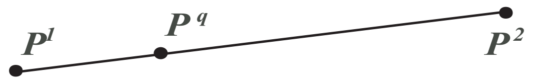


Figure 2.6: Points on a line.

The general equation for P^q is given by

$$P^q = tP^2 + (1-t)P^1, \tag{2.3.1}$$

[3]The Perron matrix is the matrix obtained by adding an identity matrix of equal dimension to a Laplacian matrix. Its row elements sum to 1.

where $t \in [0,1]$. For the two-dimensional case of (2.3.1), the matrix form is

$$[P_x^q, P_y^q] = [(1-t) \quad t] \begin{bmatrix} P_x^1 & P_y^1 \\ P_x^2 & P_y^2 \end{bmatrix}. \tag{2.3.2}$$

Let $\mathbf{H} = [(1-t) \quad t]$ be a plant or transformation matrix for system (2.3.2), then P^q is determined by choosing a value for t. For example, let $t = 0.01, (1-t) = 0.99,\ P^1 = (6.45, 4.81),\ P^2 = (7.78, 5.98)$, then

$$\begin{aligned} [P_x^q, P_y^q] =& [0.99 \quad 0.01] \begin{bmatrix} 6.45 & 4.81 \\ 7.78 & 5.98 \end{bmatrix}, \\ =& [6.58 \quad 4.92]. \end{aligned}$$

Let us extend the formulation to four points whose connecting lines form a closed four sided polygon. Suppose for each line, one wants to find a point between its two endpoints (see Figure 2.7). Then following (2.3.2), the solution is given in equation (2.3.3).

$$\begin{bmatrix} P_x^1(k+1) & P_y^1(k+1) \\ P_x^2(k+1) & P_y^2(k+1) \\ P_x^3(k+1) & P_y^3(k+1) \\ P_x^4(k+1) & P_y^4(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} (1-t) & t & 0 & 0 \\ 0 & (1-t) & t & 0 \\ 0 & 0 & (1-t) & t \\ t & 0 & 0 & (1-t) \end{bmatrix}}_{\mathbf{H}} \begin{bmatrix} P_x^1(k) & P_y^1(k) \\ P_x^2(k) & P_y^2(k) \\ P_x^3(k) & P_y^3(k) \\ P_x^4(k) & P_y^4(k) \end{bmatrix}. \tag{2.3.3}$$

Replacing the P points with a matrix $\mathbf{X}$, then (2.3.3) can be written in compact vector form as

$$\mathbf{X}(k+1) = \mathbf{H}\mathbf{X}(k).$$

It is simple to observe that the matrix $\mathbf{H}$ satisfies the following properties:

(i) The rows of $\mathbf{H}$ sum to 1.

(ii) $\mathbf{H}$ is a doubly stochastic[4] stochastic nonnegative matrix with a trivial eigenvalue of 1.

(iii) The eigenvalues of $\mathbf{H}$ are in the unit circle on the complex plane.

[4]A nonnegative matrix is doubly stochastic if its row and column elements sum to 1.

(iv) $\mathbf{H}$ satisfies $\mathbf{H1} = \mathbf{1}$, $\mathbf{1}^T\mathbf{H} = \mathbf{1}^T$,

where $\mathbf{1}^T$ is a column vector whose elements are 1.

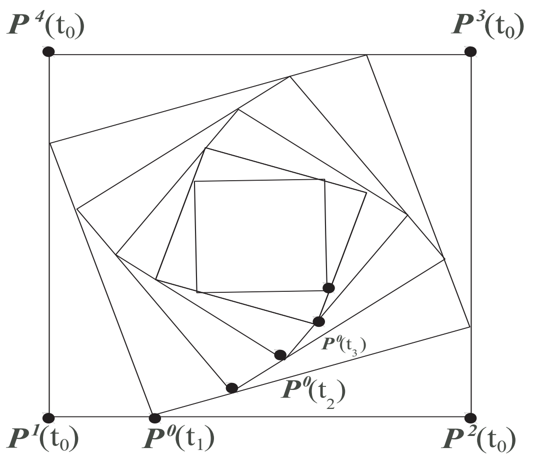


Figure 2.7: The evolution of the points on a line with time.

Theorem 2.2. *Let $\mathcal{G}$ be the directed graph represented by the outer polygon in Figure 2.7. Each vertex P^i sees the next P^{i+1}, with the last P^4 seeing P^1. Then matrix* $\mathbf{H}$ *above is the Perron matrix of $\mathcal{G}$, i.e. the matrix obtained by solving the matrix exponential of the Laplacian $\mathcal{L}$ of $\mathcal{G}$.*

Proof: The proof follows from the derivation of $\mathbf{H}$ from the circulant graph depicted by Figure 2.7. Clearly, $\mathbf{H}$ is a circulant matrix, and also a Perron matrix.

Theorem 2.2 shows that the dynamics of a circulant Laplacian matrix leads to convergence via cyclic motion. In other words, a dynamic system, or a consensus protocol that has a circulant Laplacian plant matrix is stable.

The next step is to extend the notion to a more general situation. To begin, consider three points P^1, P^2, P^3 in space, as shown in Figure 2.8 (a). Suppose one wants to find a point P^q which is on the line segment between point P^1 and a point, say P^k that is on the line segment between P^2 and P^3. One may take the midpoint of the segment between P^2 and P^3, say $P^k = P^{2.5}$, and determine P^q on the line segment between P^1 and P^k, as shown in Figure 2.8 (a). Furthermore, suppose one wants to find P^q on a line between P^1 and the centroid of a constellation of points $P^2 \dots P^n$, as shown in Figure 2.8 (b).

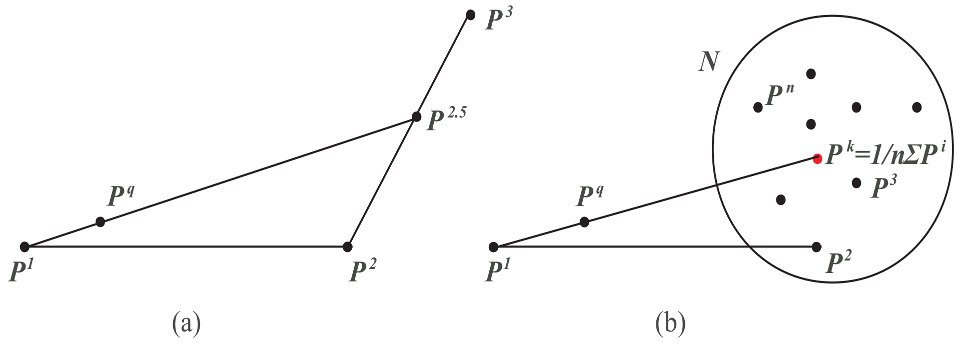


Figure 2.8: Points on a line, the evolution of the point P^q.

For simplicity, let us consider the case of three points converging to their centroid. First one obtains P^k by expanding (2.3.2) to obtain

$$P^k = [(1-t)P_x^2 + tP_x^3 \quad (1-t)P_y^2 + tP_y^3]. \tag{2.3.4}$$

Next, to obtain P^q

$$\begin{aligned} P^q &= [(1-t) \quad t] \begin{bmatrix} P_x^1 & P_y^1 \\ P_x^k & P_y^k \end{bmatrix} \\ &= [(1-t)P_x^1 + tP_x^k \quad (1-t)P_y^1 + tP_y^k]. \end{aligned} \tag{2.3.5}$$

Substituting (2.3.4) into (2.3.5) gives

$$\begin{aligned} P^q &= [(1-t)P_x^1 + t((1-t)P_x^2 + tP_x^3) \quad (1-t)P_y^1 + t((1-t)P_y^2 + tP_y^3)] \\ &= [(1-t)P_x^1 + t(1-t)P_x^2 + t^2P_x^3 \quad (1-t)P_y^1 + t(1-t)P_y^2 + t^2P_y^3)]. \end{aligned}$$

In compact matrix form:

$$P^q = [(1-t) \quad t(1-t) \quad t^2] \begin{bmatrix} P_x^1 & P_y^1 \\ P_x^2 & P_y^2 \\ P_x^3 & P_y^3 \end{bmatrix}.$$

With the choice of $t = 0.5$, and randomly selected points, $P^1 = [1.4 \quad 1.54]$, $P^2 = [4.93 \quad 2.15]$ and $P^3 = [3.54 \quad 3.46]$, one has $P^q = [2.81 \quad 2.17]$.

Now suppose also that P^2 is pursuing a point on the line segment between itself and the mid point of P^1 and P^3, while P^3 is pursuing a point on the line segment between itself and the mid point of P^1 and P^2. The three points will iteratively converge to the centroid of their initial positions in finite time. The solution is given by

$$\begin{bmatrix} P_x^1(k+1) & P_y^1(k+1) \\ P_x^2(k+1) & P_y^2(k+1) \\ P_x^3(k+1) & P_y^3(k+1) \end{bmatrix} = \begin{bmatrix} (1-t) & t(1-t) & t^2 \\ t^2 & (1-t) & t(1-t) \\ t(1-t) & t^2 & (1-t) \end{bmatrix} \begin{bmatrix} P_x^1(k) & P_y^1(k) \\ P_x^2(k) & P_y^2(k) \\ P_x^3(k) & P_y^3(k) \end{bmatrix}. \tag{2.3.6}$$

By induction, (2.3.6) can be extended to n points, yielding

$$\underbrace{\begin{bmatrix} P_x^1(k+1) & P_y^1(k+1) \\ P_x^2(k+1) & P_y^2(k+1) \\ \vdots & \vdots \\ P_x^n(k+1) & P_y^n(k+1) \end{bmatrix}}_{\mathbf{X}(k+1)} = \underbrace{\begin{bmatrix} (1-t) & t(1-t) & \cdots & t^{n-1}(1-t) & t^n \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ t^{n-1}(1-t) & t^n & \cdots & t(1-t) & (1-t) \end{bmatrix}}_{\mathbf{H}(k)} \underbrace{\begin{bmatrix} P_x^1(k) & P_y^1(k) \\ P_x^2(k) & P_y^2(k) \\ \vdots & \vdots \\ P_x^n(k) & P_y^n(k) \end{bmatrix}}_{\mathbf{X}(k)}. \tag{2.3.7}$$

Let us perform some algebraic manipulation on (2.3.7),

$$\begin{aligned} \mathbf{X}(k+1) &= (\mathbf{I} + \underbrace{(\mathbf{H} - \mathbf{I})}_{\mathbf{L}})\mathbf{X}(k), \\ \mathbf{X}(k+1) &= \underbrace{\mathbf{I}}_{\mathbf{A}}\mathbf{X}(k) + \mathbf{B}\underbrace{\mathbf{L}\mathbf{X}(k)}_{-\mathbf{u}(k)}, \\ \mathbf{X}(k+1) &= \mathbf{A}\mathbf{X}(k) - \underbrace{\mathbf{B}}_{-\mathbf{I}}\mathbf{u}(k), \\ \mathbf{X}(k+1) &= \mathbf{A}\mathbf{X}(k) + \mathbf{B}\mathbf{u}(k), \end{aligned} \tag{2.3.8}$$

where **I** is an identity matrix. Note that the matrix **L** satisfies the following properties:

(i) The rows of **L** sum to 0.

(ii) **L** is a nonnegative matrix with a trivial eigenvalue of 0.

(iii) The eigenvalues of **L** are in the unit circle.

(iv) $\mathbf{L}$ satisfies $\mathbf{L1} = \mathbf{0}$, $\mathbf{1}^T\mathbf{L} = \mathbf{0}^T$, where $\mathbf{0}$ is a column vector whose elements are all 0. Therefore $\mathbf{1}$ or any vector of uniform elements is in the nullspace of $\mathbf{L}$.

Theorem 2.3. *Let $\mathcal{G}$ be the undirected graph, or fully connected digraph, whose vertices are the points $P^1, \cdots, P^n$ in Figure 2.8 (b), with every P^i pursuing the centroid of every $P^j, \cdots, P^n$ $(P^i \notin (P^j, \cdots, P^n))$. Then matrix $\mathbf{L}$ is the Laplacian matrix of $\mathcal{G}$, and equation (2.3.7) iteratively drives the points to convergence to the centroid of their initial positions.*

Proof: The proof follows from the derivation of $\mathbf{L}$ from the graph represented by Figure 2.8. In fact, the last paragraph in p. 32 and the second paragraph in p. 34, put together, describe a fully connected digraph.

This derivation shows that if the communication graph of a team of communicating vehicles is strongly connected, convergence is guaranteed. The proof of Theorem 2.3 shows that the consensus protocol (1.3.2) is convergent.

Corollary 2.2. It follows that $\mathbf{u}$ in (2.3.8) is the consensus protocol (1.3.2).

Corollary 2.2 can be observed in the extraction of $\mathbf{u}$ from $\mathbf{H}$ in 2.3.8.

Corollary 2.3. Any nonnegative Laplacian-like stochastic matrix whose eigenvalues are in the unit circle drives a set of points to consensus.

Corollary 2.3 can be observed in the two sets of properties or rules characterising the plant matrices obtained in the derivations.

2.4 Concluding Remarks

The highlight of this chapter is the derivation of the consensus protocol presented in Section 2.3. The aim is essentially to show that any nonnegative stochastic matrix whose eigenvalues are in the unit circle drives a set of points to a consensus value. For **L**, apart from the zero eigenvalue the other eigenvalues are positive but may be much larger than one. Usually condition (iii) on p. 32 is imposed by scaling **L** by an input bias which is a small positive number ϵ $(0 < \epsilon \ll 1)$. If this scaling is not imposed the control generated is usually unbounded thus making the system unstable. Therefore it is safe to state that the eigenvalues of **L** are in the unit circle because the system can be scaled. Therefore, for a dynamic system whose plant matrix is a Laplacian-like stochastic matrix, the proof of consensus is equivalent to showing that the plant matrix is indeed a nonnegative stochastic matrix whose eigenvalues are in the unit circle.

Chapter 3

Consensus Based Multi-Path Planning With LMI Avoidance Constraints

One of the central problems of path planning is that of collision avoidance, and one application of consensus theory is in cooperative path planning. In this chapter, a new approach to the general problem of collision avoidance is developed, and used to incorporate collision avoidance into the consensus algorithm. The collision avoidance approach is based on the concept of quadratically constrained attitude control (Q-CAC), where attitude constraints are represented as a set of linear matrix inequalities (LMI), and solved by semidefinite programming (SDP). The approach is used to simulate consensus based multi-path planning with collision avoidance, for a team of communicating UAVs, and soccer robots. A stability analysis is also presented, together with the simulation results, to demonstrate the effectiveness of the approach (Okoloko and Basson, 2011; Okoloko, 2012*b*).

3.1 Introduction

The large volume of literature on the subject of multi-path planning with collision avoidance is justified by the current and future potential applications of path planning, which include coordinated control of: (i) multiple robot systems (Zickler *et al.*, 2009), and multiple vehicles in intelligent highways (PATH, 2006); (ii) multiple unmanned aerial and space systems (Chandler *et al.*, 2001; Richards *et al.*, 2002; Kim *et al.*, 2004; Keviczky *et al.*, 2008; Okoloko and Kim, 2010); (iii) mobile sensor network systems (Blackwood *et al.*, 2002); (iv) environmental sampling systems (MBARI, 2006); and (v) air, sea and land traffic control systems (Hwang and Tomlin, 2002; Hu, 2003).

Some researchers have approached path planning with collision avoidance as a constrained optimization problem, where the objective function is to optimally reach desired goal positions, subject to constraints such as: initial and final positions and velocities; dynamics; collision avoidance; and velocity bounds (Richards *et al.*, 2002; Kim *et al.*, 2004; Keviczky *et al.*, 2008; van den Berg *et al.*, 2009). Keviczky *et al.* (2008) incorporated consensus into the mixed integer linear programming (MILP) framework to enable formation manoeuvres. The general drawback of optimization based approaches is that the complexity increases with the number of vehicles, with the result that only a limited number of vehicles can be controlled in real-time.

Among many others, Hwang and Tomlin (2002) proposed a protocol based conflict resolution for safe path planning for multiple aircraft, which is also similar to the work of Hu (2003) in air traffic control. However, for effective collision avoidance, their formulation requires that each aircraft knows the exact position, velocity and heading of every other aircraft at all times. While it is possible to obtain such information, the availability of such information cannot always be guaranteed at all times due to the limitations of wireless communications. As such, it is desirable to also give consideration to reactive strategies, where one aircraft (or vehicle) can estimate only the position and orientation of any other aircraft (or vehicle) intersecting its safety region at any time. Moreover, consensus algorithms that enable arbitrary formation reconfiguration manoeuvres with avoidance, using minimal control efforts, were not included in the framework of Hwang and Tomlin (2002).

The simplicity and immense potential applications of the consensus algorithm has made it an attractive choice for multi-agent control. For the algorithm to be practically useful in the control of multiple vehicles, collision avoidance is necessary. Many researchers have tried to handle the avoidance problem by introducing potential forces such as attraction and repulsion. See for example Section I in the paper of Peng *et al.* (2009) for a list of references. As an example, a potential function based avoidance algorithm was proposed by Olfati-Saber (2006) for collision free flocking. However, the flocking algorithms were not developed for arbitrary reconfigurations, e.g. vehicles moving in opposite directions. One general drawback of potential function algorithms is that of getting into local minima. Also, it is observed by Peng *et al.* (2009) that any repulsions based on potential functions alone are not sufficient to guarantee consensus based collision avoidance.

To eliminate collisions in consensus, Peng *et al.* (2009) proposed a model with adaptive speed. The maximum speed of the i^{th} vehicle is set at $v^i = (d^i - 2a)/2$, where d^i is the Euclidean distance between the two centers of the i^{th} vehicle and its nearest neighbour, and a is the diameter of the vehicles. An attitude change manoeuvre is also included whenever collisions are imminent. However, the model requires each vehicle to know the position, velocity and heading of all its close neighbours. Consider a practical situation in which some large external perturbation forces one vehicle to move too closely to another vehicle, as may be common in densely packed populations. Then at some point $d^i \approx 2a$, and a stalemate situation is possible. The formulation in Peng *et al.* (2009) resulted in situations where the speed of the overall team can be greatly reduced, even when the direction of motion will not lead to collisions. A scattering model was proposed to overcome this, but it also resulted in undesirable abrupt phase transitions. The slow speed of convergence is cited in Peng *et al.* (2009) as a limitation of this model. Accordingly, designing efficient methods to simultaneously guarantee the absence of collisions and fast convergence remains an open problem. Moreover, the attitude change manoeuvre presented by Peng *et al.* (2009) was not developed for three dimensional attitude manoeuvres.

Thus, in this work, a new approach to incorporating collision avoidance into the consensus framework using quadratically constrained attitude control (Q-CAC), via semidefinite programming (SDP), using linear matrix inequalities (LMI), is proposed. Several benefits of this approach, and differences from other approaches, are: (i) each vehicle only needs to know the relative distance and direction of motion of other vehicles within its minimum safety distance; not necessarily their exact positions, velocities and orientations as in the works of Hwang and Tomlin (2002), Hu (2003), and Peng *et al.* (2009) and this information can be obtained from long range onboard sensors on each vehicle, even when networked communication systems fail; (ii) the formulation can be applied to two dimensional as well as three dimensional consensus with collision avoidance, without any further modifications; (iii) computational complexity is reduced because the complete problem is not solved as an optimization problem every time step as in the works of Richards *et al.* (2002), Kim *et al.* (2004) and Keviczky *et al.* (2008), so a larger number of vehicles can be controlled in real-time. In addition, in this work, a consensus algorithm is developed to enable reconfiguration to exact desired positions.

The contributions of this work are therefore stated as follows: (i) a consensus based algorithm for reconfiguration to desired final locations is developed; (ii) a new method is developed for collision avoidance by using the concept of Q-CAC; (iii) a mathematical proof of consensus is developed.

The rest of the chapter is organized as follows. In Section 3.2 brief mathematical preliminaries are presented for the problem considered. The problem is formulated in Section 3.3, and the solution technique and convergence analysis are provided in Section 3.4. Numerical simulations are given in Section 3.5, and concluding remarks follows in Section 3.6.

3.2 Consensus Theory Revisited

The consensus algorithm (1.3.2) introduced in Section 1.3 will be re-stated here in a graph theoretic parlance. The *consensus* problem is that of driving the states of a team of communicating agents to a common value, by distributed protocols based on their communication graph. The agents are represented by vertices of the graph, while the edges of the graph represent communication links between them. Let x^i denote the state of a vehicle i $(i = 1, \cdots, n)$. For systems modelled by first-order dynamics, the following first-order consensus protocol (or its variants) have been proposed (e.g. Olfati-Saber and Murray, 2004; Ren *et al.*, 2005)

$$\dot{x}^i(t) = -\sum_{j \in \mathcal{N}_i} b^{ij} a^{ij}(t)(x^i(t) - x^j(t)), \tag{3.2.1}$$

where a^{ij} is an element of the adjacency matrix, and b^{ij} $(0 < b^{ij} \ll 1)$ is an input bias which is used to bound the outputs to ensure stability. Dropping b^{ij} for simplicity, the collective dynamics of $(3.2.1)$ is

$$\dot{\mathbf{x}}(t) = -\mathbf{L}\mathbf{x}(t), \tag{3.2.2}$$

where $\mathbf{L}$ is the Laplacian matrix of the communication graph. A sufficient condition for consensus to be achieved using protocol $(3.2.2)$ is for the graph to contain a directed spanning tree for each time $t \in [t_0, t_f]$ (Ren *et al.*, 2005), where t_0 and t_f are the initial and final times. Assuming this condition holds, protocol $(3.2.2)$ solves the rendezvous problem. In its more general form, by augmenting $(3.2.2)$ with a relative offset vector $\mathbf{x}^{off}$, it also solves the formation control problem. The protocol is given as

$$\dot{\mathbf{x}}(t) = -\mathbf{L}(\mathbf{x}(t) - \mathbf{x}^{off}). \tag{3.2.3}$$

By the definition of the Laplacian, $\mathbf{L}$ always has a zero eigenvalue $\lambda_1 = 0$, corresponding to the eigenvector $\mathbf{1} = [1 \cdots 1]^T$, because $\mathbf{1}$ belongs to the null space of $\mathbf{L}$, i.e. $\mathbf{L1} = \mathbf{0}$. This implies that an equilibrium state of $(3.2.2)$ is a state $\mathbf{x}^* = [\alpha \cdots \alpha]^T = \alpha\mathbf{1}$, in which all vehicles agree, to the consensus value $\alpha = \frac{\mathbf{1}^T}{n}\mathbf{x}(0)$. From this point on, $\|x^i - x^j\| \to 0$. Therefore consensus is said to have been achieved when $\|x^i - x^j\| \to 0$ for $(3.2.2)$ or $\|x^i - x^j\| \to (x^{ij})^{off}$ for $(3.2.3)$, as $t \to \infty$, $\forall\ i \neq j$.

3.3 Problem Statement

The canonical multi-path planning problem is re-stated here for clarity: Given a set of vehicles i $(i = 1, \cdots, n)$, with initial positions $x^i(t_0)$ at time t_0, desired final positions x^i_d, at time t_f, a set of obstacles with positions x_{obs}^j $(j = 1, \cdots, m)$, and the Laplacian matrix of their communication graph $\mathbf{L}$, find a sequence of collision free trajectories from t_0 to t_f such that $x^i(t_f) = x_d^i$ for all i.

Part of this problem is a formation control problem, which can be solved by using protocol (3.2.3). However, it is beset with some extra problems. First, for the formation acquisition part, the centroid of formation achieved using (3.2.3) converges to the centroid of the initial vehicle positions, and not necessarily to desired positions, because $\mathbf{L}$ is an attractor matrix, and this property was shown in Section 2.3. If a leader-follower graph Laplacian with a single leader is employed, then the vehicles converge to a formation relative to the static leader vehicle and, again, not to desired final positions. Secondly, there is the need to incorporate reliable collision avoidance into (3.2.3) which also guarantees a high speed of convergence.

3.4 Solution

In this section, we develop a solution to the problem stated in Section 3.3. The solution involves four intermediate steps: (i) consensus based reconfiguration to desired positions; (ii) formulation of Q-CAC based collision avoidance; (iii) conflict resolution for multiple vehicles; (iv) consensus with Q-CAC based collision avoidance.

3.4.1 Consensus Based Reconfiguration to Desired Positions

It was previously shown by Fax (2002) that for the dynamic system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u},$$

there exist stabilizing feedback controllers $\mathbf{F}$, such that the protocol

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{F}\mathbf{u}$$

drives $\mathbf{x}$ to $\mathbf{x}_f$, where $\mathbf{x} = [x^1, \cdots, x^n]$ is a stacked vector of the initial positions of the vehicles, $\mathbf{u} = -\Gamma(\mathbf{x} - \mathbf{x}^{off})$, $\Gamma = \mathbf{L} \otimes \mathbf{I}_p$, $\mathbf{I}_p$ is the identity matrix of size $p \times p$, and p is the state dimension of the vehicles. Further, Lafferriere *et al.* (2005) proves that there exists a relationship between the **F** which guarantees convergence to a formation, and the communication graph $\mathcal{G}$ of **L**. However, in this work, rather than combining path planning with tracking control, the reference consensus path planning problem is considered first. To this end, the following protocol is proposed for a leader-follower architecture:

$$\mathbf{u} = -\Gamma(\mathbf{x} - \mathbf{x}^{off}) + \mathbf{K}(\mathbf{x}^{off} - \mathbf{x}). \tag{3.4.1}$$

The corresponding protocol for a leaderless architecture is:

$$\mathbf{u} = -\Gamma(\mathbf{x} - \mathbf{x}^{off}) + \mathbf{K}(\mathbf{x}_d - \mathbf{x}), \tag{3.4.2}$$

where $\mathbf{x}_d \neq \mathbf{x}^{off}$ is the desired final position and is different from the formation configuration, $\mathbf{K} = \epsilon\mathbf{I}_n$, $(0 < \epsilon \ll 1)$, and n is the dimension of **x**.

Theorem 3.1 *The time varying system* $(3.4.1)$ *achieves consensus.*

Proof: Without loss of generality, assume that $\mathbf{x}^{off} = \mathbf{0}$ is a vector consisting of all zeros. If $\Lambda = \Gamma + \mathbf{K}$, one can write $(3.4.1)$ as

$$\mathbf{u} = \dot{\mathbf{x}} = -\Lambda\mathbf{x}.$$

Since Γ is a Laplacian matrix and **K** is diagonal and positive definite, the addition of Γ and **K** shifts the eigenvalues Γ by the positive number ϵ, therefore the positive semidefinite matrix Γ is transformed to a positive definite matrix Λ. By virtue of the structure of the Laplacian matrix Γ, the matrix Λ is a nonnegative stochastic matrix satisfying the following properties.

(i) The elements in every row of Λ sum to ϵ.

(ii) Λ is a nonnegative stochastic matrix with a trivial eigenvalue of ϵ.

(iii) The eigenvalues of Λ are in a unit circle.

(iv) Λ satisfies $\Lambda\mathbf{1} = \mathbf{k}$; $\mathbf{1}^T\Lambda = \mathbf{k}^T$,

where $\mathbf{k}^T$ is a column vector all of whose elements are ϵ. By Corollary 2.3, the matrix Λ drives the set of points to consensus. This proves the claim.

Topology 1 shown in Figure 3.1 is a leader-follower graph. In the figure, node 1 is the leader and each of the other nodes are connected to their adjacent neighbours. The result of applying protocol $(3.4.1)$ with Topology 1, to the simulation of reconfiguration of 11 soccer robots in a soccer play-off formation, is shown in Figure 3.2. In the figure, the dots inside small circles indicate initial positions, while the dot in the diamond is the initial position of the leader. The stars indicate desired final positions. The larger circles with dashed lines are positions where collisions occurred, and the diameters of the circles indicate the size of intersection of the safety regions of the vehicles. In Figure 3.3, protocol $(3.4.1)$ is applied with the same topology to the simulation of reconfiguration of 12 UAVs visiting three regions. In the experiment, the UAVs are initially randomly positioned about a region ($REGION_1$), then they fly to converge to a circular arc formation around a region of interest ($REGION_2$), after that they fly to converge to a circular formation about a third region of interest ($REGION_3$). Finally, they fly back to their initial positions. In the figure, the small black sphere following the black trail is the leader UAV. The thick black blurs along the trajectories indicate positions where collisions occurred. Note that the leader UAV is undertaking a lane change manoeuvre in the first lap.

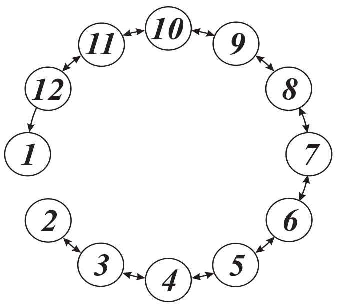


Figure 3.1: Topology 1: a leader-follower graph.

In Figure 3.2, the curvature path shown by Vehicles 5 and 8 even though they are close to their destinations, is due to the use of a circulant Laplacian communication matrix in controlling their motion. The vehicles do not know their final destination, only vehicle 1 (the goal keeper) knows; the vehicles were moving by communicating with vehicle 1. Using a fully connected Laplacian, the vehicles approach their goals in a straight line. In future development it is appropriate to include logical rules, such as disconnecting a vehicle from communication that affects its motion, after it already knows its goal or is close to its goal, thereby avoiding such looping behaviour even when the communication graph is circulant.

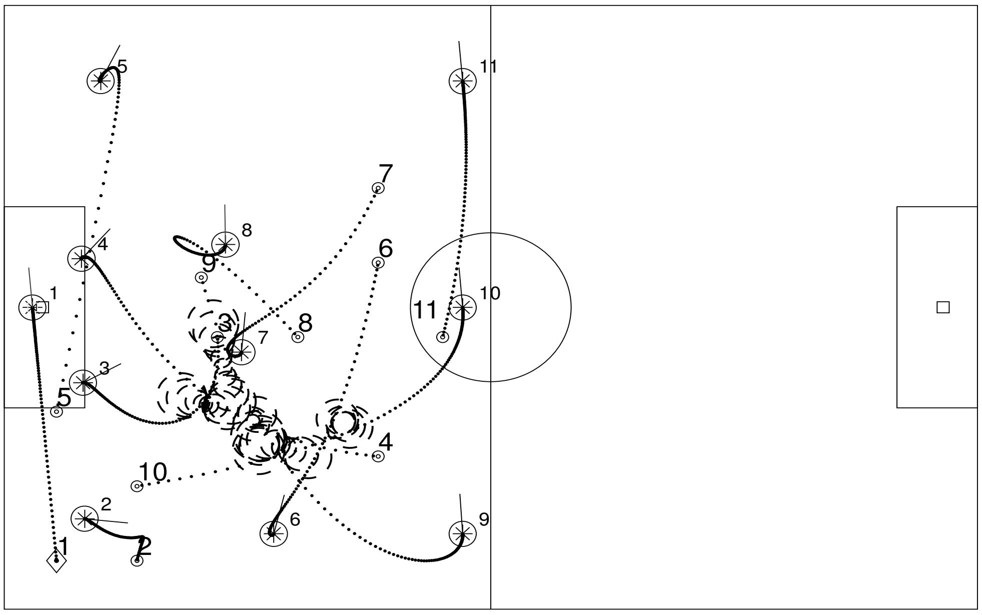


Figure 3.2: Reconfiguration of eleven soccer robots to exact desired positions using Topology 1.

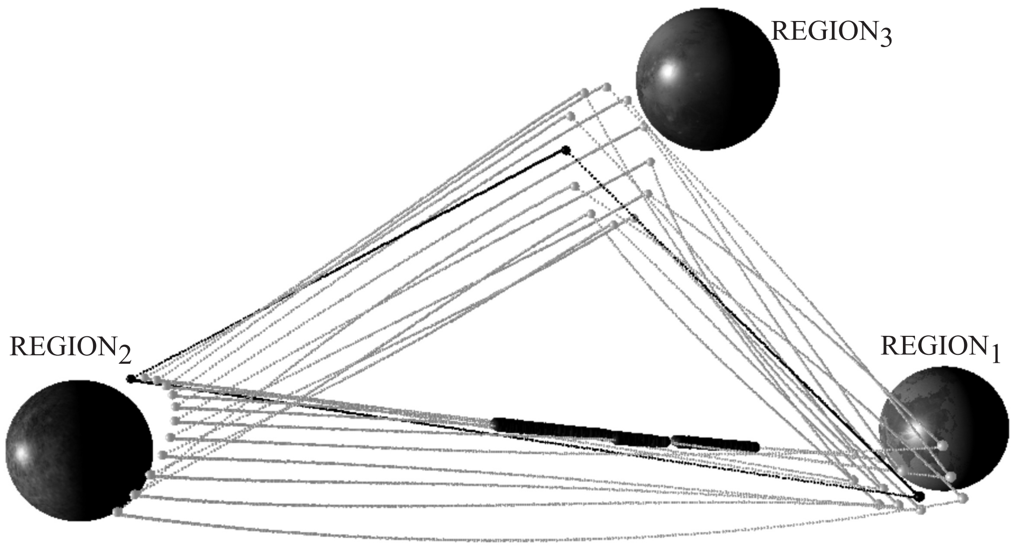


Figure 3.3: Reconfiguration of twelve UAVs to exact desired formation positions using Topology 1.

3.4.2 Q-CAC Based Collision Avoidance

The collision avoidance problem is that of driving the state of a vehicle from one point to another, while avoiding static obstacles and collisions with other moving vehicles. For simplicity, we approximate a vehicle or an obstacle by **S**, as shown in Figure 3.4. A non-spherical obstacle may be represented by a polygon as shown in Figure 3.5. For the **S** type obstacle (or vehicle), if the obstacle is centred on x_{obs}, it is desired that the time evolution of any vehicle state $x^i(t)$ from t_0 to t_f should avoid the constraint region shown in Figure 3.4.

The feasible region is thus defined by

$$x_{feas} = \left\{x \in \mathbb{R}^{m \times m} |\, \|x - x_{obs}\| > r^*\right\},\ m \in \mathbb{R}, \tag{3.4.3}$$

where r^* is the radius of **S**[1], bounded by a safety region of width $\mathcal{E}$.

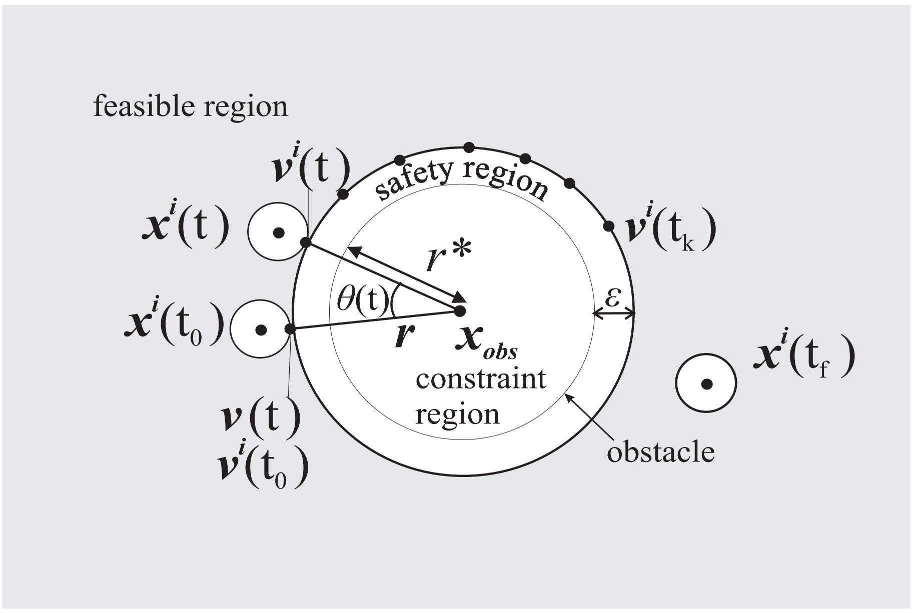


Figure 3.4: Constrained control problem for a static spherical obstacle.

Although the nonlinear nonconvex equation (3.4.3) does not have a direct representation as LMI, some non-LMI methods, e.g. MILP (Richards *et al.*, 2002; Keviczky *et al.*,

[1]Or the length of the longest line segment from the centroid of a non **S** type obstacle to a point on the boundary of the obstacle.

2008), have been developed for approximating its solution. In this section, we propose a new approach based on the Q-CAC algorithm that was initially developed by Kim and Mesbahi (2004) for the attitude control problem.

At any time t, suppose the safety region of vehicle i centered on $x^i(t)$ intersects the safety region of an obstacle, *obs*, centered on x_{obs}. Let $v(t)$ be the unit vector extending from the centre of x_{obs} or $x^i(t)$ in the direction of the point of intersection. The vectors $v(t)$ will be different for each vehicle or obstacle. Consider the case shown by Figure 3.4, suppose x_{obs} is known, and $v(t)$ is also known in the frame of *obs*. Then, to guide vehicle i safely around the obstacle, define a unit vector $v^i(t)$ in the direction of $v(t)$ in the frame of obs. The vector $v^i(t)$ will be regarded as an imaginary vector whose direction can be constrained to change with time. The vector $v^i(t)$ can then be used to find a sequence of trajectories around *obs* which guides i from $x^i(t_0)$ to $x^i(t_f)$ without violating (3.4.3).

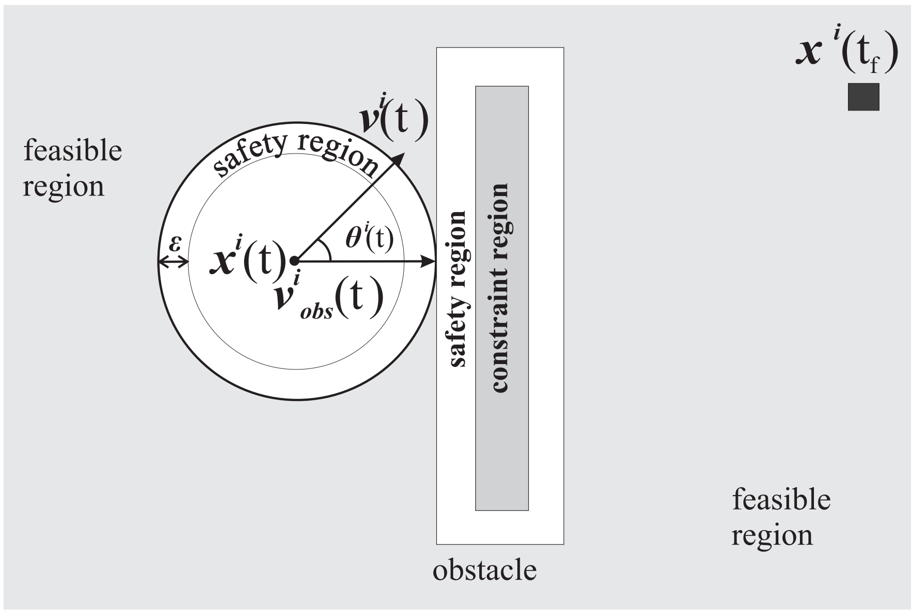


Figure 3.5: Constrained control problem for static nonspherical obstacle.

This problem reduces to the Q-CAC problem. It is desired that the angle θ between $v^i(t)$ and $v(t)$ should be larger than some given angle ϕ, $\forall\, t$. The constraint is

$$v^i(t)^T v(t) \leq \cos\phi, \;\; \forall t \in [t_0, t_f]. \tag{3.4.4}$$

The idea is to expand the angle between the unit vectors $v^i(t)$ and $v(t)$. This implies that one of the vectors $v^i(t)$ or $v(t)$ must remain static, while the other moves with time. Vector $v^i(t)$ is used to control the position of the vehicle, therefore $v^i(t)$ will move with time, the positions of $v^i(t)$ defines a trajectory path for $x^i(t)$. Thus, $x^i(t)$ is forced to move on the surface of the safety region bounding **S**. At some time t_k, $x^i(t)$ will arrive close to a point indicated by $v^i(t_k)$, at which a translation to $x^i(t_f)$ is unconstrained. This is shown by the black dots on the boundary of the safety region in Figure 3.4. To obtain the unit vector $v(t)$, the actual vector extending from the centre of x_{obs} or $x^i(t)$ in the direction of the point of intersection is normalised. After the solution $v^i(t)$ is obtained as a unit vector, $v^i(t)$ is multiplied by $r = r^* + \mathcal{E}$ in order to obtain the actual safe trajectory.

Let $\mathbf{v}(t) = [v^i(t)^T \; v(t)^T]^T$, then the dynamics of $\mathbf{v}(t)$ is defined as

$$\dot{\mathbf{v}}(t) = \mathbf{D}(t)\mathbf{v}(t),$$

where $\mathbf{D} \in \mathcal{S}^{pn}$, p is the dimension of the state vector x^i, n is the number of vehicles. The above differential equation represents the rotational dynamics of the two vectors contained in $\mathbf{v}(t)$. **D** is a semidefinite matrix variable whose contents are unknown. Its purpose is to expand the angles between the two vectors in $\mathbf{v}(t)$, while also keeping them normalized.

The discrete time equivalent of the above differential equation is

$$\mathbf{v}(k+1) = \Delta t\mathbf{D}(k)\mathbf{v}(k), \tag{3.4.5}$$

where $k = 0, \cdots, N$ $(N\Delta t = t_f)$ is the discrete time equivalent of t, and Δt is the discretization time step. To implement (3.4.5), **D** is declared in a semidefinite program which chooses the appropriate values to rotate the vectors in $\mathbf{v}(t)$ while satisfying norm constraints. Note in the above discretization of the differential equation, the identity matrix cannot be added in this solution; instead the matrix **D** is chosen implicitly to satisfy the rotation. The vectors in $\mathbf{v}(t)$ are unit vectors; they are not translating, but they are rotating and must be preserved as unit vectors.

To enforce the constraint in (3.4.4)[2] in a SDP, it has to be represented as a LMI by using the Schur complement formula described by Boyd *et al.* (1994). The Schur complement

[2]Equation (3.4.5) enables us to define attitude constraints.

formula states that the inequality

$$\mathbf{S}\mathbf{R}^{-1}\mathbf{S}^T - \mathbf{Q} \le 0;\ \mathbf{Q} = \mathbf{Q}^T,\ \mathbf{R} = \mathbf{R}^T,\ \mathbf{R} > 0,$$

is equivalent to, and can be represented by the linear matrix inequality,

$$\begin{bmatrix} \mathbf{Q} & \mathbf{S} \\ \mathbf{S}^T & \mathbf{R} \end{bmatrix} \ge 0.$$

Note that $(3.4.4)$ is equivalent to

$$\underbrace{\begin{bmatrix} v^i(t)^T & v^T(t) \end{bmatrix}}_{\mathbf{v}^T(t)} \begin{bmatrix} \mathbf{0}_3 & \frac{1}{2}\mathbf{I}_3 \\ \frac{1}{2}\mathbf{I}_3 & \mathbf{0}_3 \end{bmatrix} \begin{bmatrix} v^i(t) \\ v(t) \end{bmatrix} \le \cos\phi, \tag{3.4.6}$$

which also implies that

$$\underbrace{\begin{bmatrix} v^i(t)^T & v^T(t) \end{bmatrix}}_{\mathbf{v}^T(t)} \underbrace{\begin{bmatrix} \mathbf{0}_3 & \mathbf{I}_3 \\ \mathbf{I}_3 & \mathbf{0}_3 \end{bmatrix}}_{\mathbf{G}} \begin{bmatrix} v^i(t) \\ v(t) \end{bmatrix} \le 2\cos\phi. \tag{3.4.7}$$

Note also that some of the eigenvalues of the $\mathbf{G}$ in (3.4.7) are non-positive. To make the matrix positive definite, one only has to shift the eigenvalues of $\mathbf{G}$, by choosing a positive real number μ which is larger than the largest absolute value of the eigenvalues of $\mathbf{G}$, then

$$\underbrace{\begin{bmatrix} v^i(t)^T & v^T(t) \end{bmatrix}}_{\mathbf{v}^T(t)} \left(\mu\mathbf{I}_6 + \begin{bmatrix} \mathbf{0}_3 & \mathbf{I}_3 \\ \mathbf{I}_3 & \mathbf{0}_3 \end{bmatrix}\right) \underbrace{\begin{bmatrix} v^i(t) \\ v(t) \end{bmatrix}}_{\mathbf{v}(t)} \le 2(\cos\phi + \mu). \tag{3.4.8}$$

If $\mathbf{M} = \left(\mu\mathbf{I}_6 + \begin{bmatrix} \mathbf{0}_3 & \mathbf{I}_3 \\ \mathbf{I}_3 & \mathbf{0}_3 \end{bmatrix}\right)^{-1}$, then $\mathbf{M}$ is positive definite. Therefore, following the Schur complement formula, the LMI equivalent of (3.4.8) is

$$\begin{bmatrix} 2(\mu + \cos\phi) & \mathbf{v}(t)^T \\ \mathbf{v}(t) & \mathbf{M} \end{bmatrix} \ge 0. \tag{3.4.9}$$

For collision avoidance, the dynamic system (3.4.5) is solved whenever it is required, subject to the attitude constraint (3.4.9), and norm constraints $\|v^i(t)\| = 1$ and $\|v(t)\| = 1$.

Thus, the optimization problem of collision avoidance is essentially to find a feasible v^i subject to the following constraints:

$$\mathbf{v}_{k+1} = \Delta t \mathbf{D}(t)\mathbf{v}_k,$$

$$\mathbf{v}_k^T(\mathbf{v}_{k+1} - \mathbf{v}_k) = 0,$$

$$\begin{bmatrix} 2(\mu + \cos\phi) & \mathbf{v}^T \\ \mathbf{v} & \mathbf{M} \end{bmatrix} \geq 0.$$

Equation $\mathbf{v}_k^T(\mathbf{v}_{k+1} - \mathbf{v}_k) = 0$ is essentially the discrete time version of $\mathbf{v}(t)^T\dot{\mathbf{v}}(t) = 0$ which guarantees that $\mathbf{v}(t)^T\mathbf{v}(t) = 2$ as long as $\left\|v^i(0)\right\| = 1$ and $\|v(0)\| = 1$.

The trajectory obtained by applying the avoidance protocol of $(3.4.5)$ to a single static obstacle avoidance problem in 2D and 3D is shown in Figure 3.6. The vehicle manoeuvres from $x(0)$ to $x(t_f)$ without violating the constraint region. This realistic scenario applies to structured environments where the centroid of the obstacle(s) is known. In this case a smooth avoidance trajectory can be obtained. If the centroid of the obstacle is unknown, a safe approximate trajectory can also be obtained, and this forms the basis for the dynamic collision avoidance case developed in this work. Figure A.8 in Appendix A also shows a practical application of the strategy to the collision free reconfiguration of one UAV avoiding a static UAV.

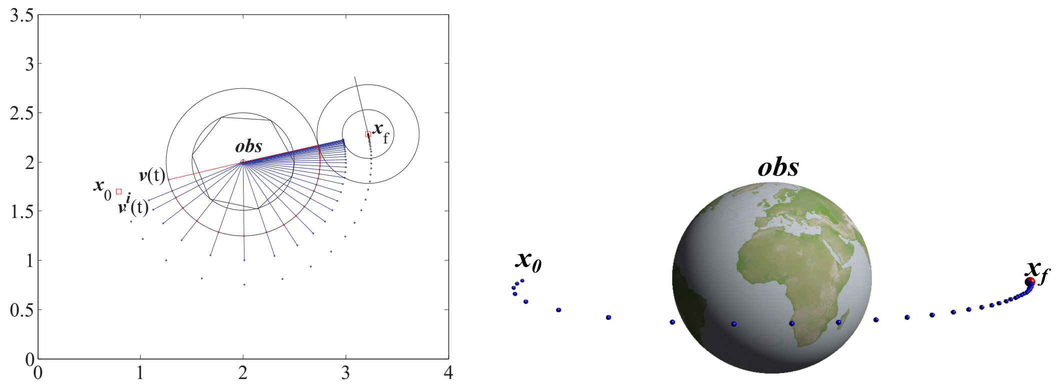


Figure 3.6: Constrained avoidance of a static obstacle.

The LMI for this problem was coded using Yalmip (Lofberg, 2004) in Matlab R2009a, and solved using Sedumi (Sturm, 1998), and it solved in $0.5s$ on an Intel(R) Core(TM)2

Duo P8600 @ 2.40GHz with 2 GB RAM, running Windows 7. It is possible to obtain a higher speed by increasing the value of Δt in (3.4.5).

Extending this formulation to the case of dynamic obstacles, consider two vehicles with states $x^i(t),\ x^j(t),\ (i \neq j)$, and attitude vectors $v^i(t),\ v^j(t)$ respectively, which must avoid each other at all times. As shown in Figure 3.7, at any time t, if any of their safety regions are violated, the point of intersection of vehicles i and j in the coordinate frame of i is $v^i_{obs}(t)$.

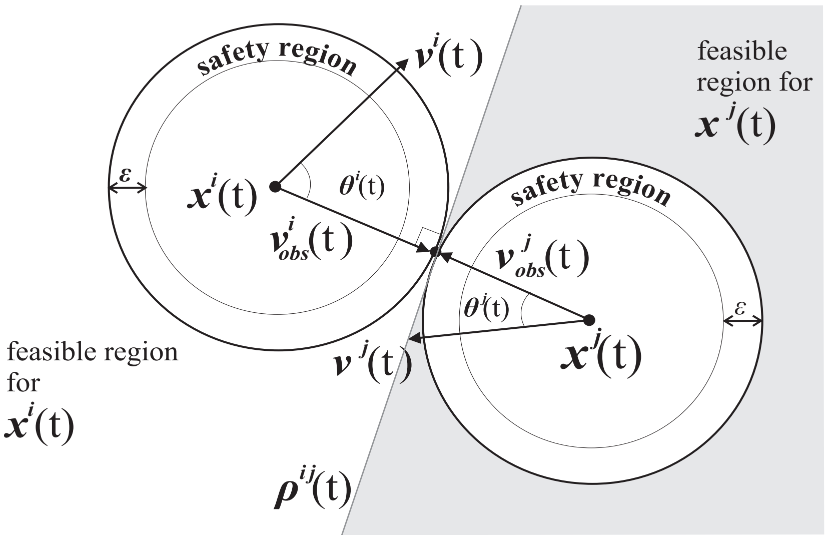


Figure 3.7: Constrained control problem for dynamic obstacles.

The avoidance requirements are

$$\theta^i(t) \geq \phi \ \equiv v^i(t)^T v^i_{obs}(t) \leq \cos\phi,$$
$$\theta^j(t) \geq \phi \ \equiv v^j(t)^T v^j_{obs}(t) \leq \cos\phi,\ \forall\, t \in [t_0, t_f]\,,$$

where $\phi \geq \pi/2$. For this dynamic situation it is sufficient to enforce the following avoidance constraints:

$$\begin{bmatrix} 2(\mu + \cos\phi) & \begin{bmatrix} v^i(k+2) \\ v^i_{obs}(k+2) \end{bmatrix}^T \\ \begin{bmatrix} v^i(k+2) \\ v^i_{obs}(k+2) \end{bmatrix} & \mathbf{M} \end{bmatrix} \geq 0, \tag{3.4.10}$$

$$\begin{bmatrix} 2(\mu + \cos\phi) & \begin{bmatrix} v^j(k+2) \\ v^j_{obs}(k+2) \end{bmatrix}^T \\ \begin{bmatrix} v^j(k+2) \\ v^j_{obs}(k+2) \end{bmatrix} & \mathbf{M} \end{bmatrix} \geq 0, \qquad (3.4.11)$$

$$i, j = 1, \cdots, n, \; i \neq j.$$

Note that $(k+2)$ is used because the optimization is performed two steps ahead of time to ensure that the future trajectories are collision free. However, when this avoidance protocol is applied to dynamic collision avoidance, some vehicle configurations pose some avoidance challenges, and this is considered next.

3.4.3 Conflict Resolution for Multiple Vehicles

A collision between two vehicles i and j is imminent at time t whenever

$$D^{ij}(t) = \left\|x^i(t) - x^j(t)\right\| \leq (r^i + r^j),$$

which can be computed by using position feedback data determined by onboard or external sensors, or communicated among the vehicles.

There are two aspects of collision problems: (i) collision detection; (ii) collision response. Collision detection is the computational problem of detecting the intersection of two or more objects. To do this numerically, it generally requires extensive use of concepts from linear algebra and computational geometry (Eberly, 2001; van den Bergen, 2004). There are commercially available and free software packages for simulating collision detection and response, for example NVIDIA's PhysX[3] which is being used by the CMU Dragons team (Zickler *et al.*, 2009). In addition to determining whether two objects have collided, such software systems may also calculate time of impact (TOI), and report a contact manifold (the set of intersecting points). Where such systems are available, they can be interfaced to a path planning algorithm in order to help with detecting collisions. Collision response is the initiation of the appropriate avoidance manoeuvre. In this section, we

[3]http://www.geforce.com/Hardware/Technologies/physx

develop methods to detect different configurations of collisions and classify them. Then an appropriate response technique is developed for each of the collision configurations.

Consider two vehicles i and j, whose current states are $x^i(t)$ and $x^j(t)$, and the desired final states are $x^i(t_f)$ and $x^j(t_f)$. We identify three different basic collision configurations as: (i) *simple collision*; (ii) *head-on collision*; (iii) *cross-path collision*. Simple solutions will be developed for these configurations and when combined synergistically, they will provide sufficient collision avoidance behaviour for fast collision free reconfiguration for the team of vehicles.

3.4.3.1 Detecting and Resolving a Simple Collision

A simple collision problem is any configuration in which $D^{ij}(t) \leq (r^i + r^j)$ and the current vector directions (or attitude vectors) $v^i(t)$ and $v^j(t)$ of the motion of vehicles i and j are on different sides of the plane or infinite line $L^{ij}(t)$ passing through the points $x^i(t)$, $x^j(t)$, and the attitude vectors $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$ are not parallel. Note that when $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$ are not parallel, a point or line of intersection can be computed for both vectors. Examples of simple collision problems are shown in Figures 3.7 and 3.8.

This is the easiest collision problem to solve because the attitude vectors are already on opposite sides of $L^{ij}(t)$. Consider Figure 3.8 (b), the plane or line $\rho^{ij}(t)$ tangent to the point of intersection of both vehicles constrains the current motion spaces of the vehicles to the two sides of the plane at time t. A pure optimization based solution will attempt to search the space on the right side of $\rho^{ij}(t)$ to seek for a point which is closest to the goal of i, and this will be used as the next trajectory. The algorithm will also search the left side of $\rho^{ij}(t)$ to find the next trajectory for j. Once the positions are updated a new $\rho^{ij}(t)$ is computed.

Indeed the solution is provided by the basic collision avoidance protocols $(3.4.10)$ and $(3.4.11)$ without having to do a set search. It is easy to observe that by expanding the angles $\theta^i(t)$ and $\theta^j(t)$ and choosing the next feasible trajectories $r^*/2$ along the new direction vectors $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$, the new trajectories are bound to satisfy the feasible regions separated by $\rho^{ij}(t)$, provided $\mathcal{E}^i \geq r^{*^i}$ for any i. The rest of the avoidance strategies developed in the remaining part of this section are attempts to reduce more complex collision configurations to a simple collision configuration.

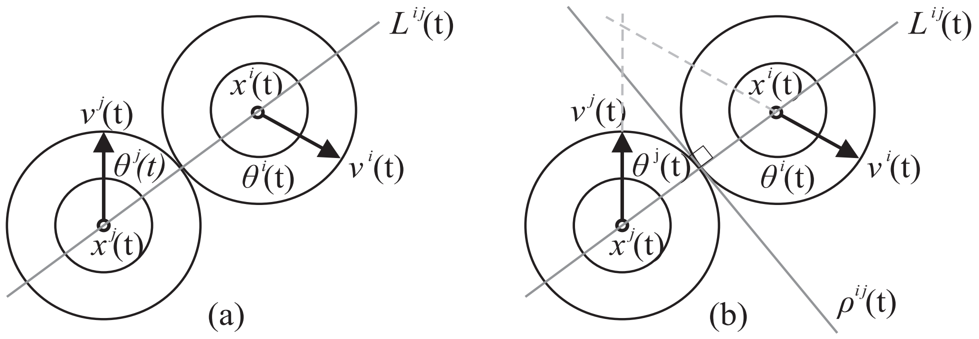


Figure 3.8: Simple collision problem.

3.4.3.2 Detecting and Resolving a Head-On Collision

A head-on collision problem is any configuration in which $D^{ij}(t) \leq (r^i+r^j)$ and $v^i(t)^T v^j(t) \approx -\pi$rad. Figure 3.9 (a) illustrates the head-on collision problem. The paths from the current positions $x^i(t)$ and $x^j(t)$ to the goal positions $x^i(t_f)$ and $x^j(t_f)$ lead to a configuration in which the attitude vectors $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$ are parallel (or close to parallel) and in opposite directions, in the sense that a point of intersection cannot be computed. Figures 3.9 (b)-(d) are several examples of head-on collision. Figure 3.9 (b) is a direct head-on collision problem because the vectors $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$ are lying directly on L^{ij}. Figure 3.9 (c) is an approximate head-on collision configuration, and Figure 3.9 (d) is a head-on collision configuration that can be easily converted to a simple collision configuration.

For the configurations in Figure 3.9 (b) and (c), the Q-CAC formulation presented earlier easily solves this problem without any modifications to the algorithm, i.e. it computes feasible avoidance trajectories. However, whenever $v^i(t)^T v^i_{obs}(t) \approx 0$ for any i, the optimization algorithm takes some significant time to solve. Even though the resulting trajectory is desirable, this delay is undesirable for real-time collision avoidance. Therefore whenever this configuration is encountered for any two vehicles, a one-step elementary evasive manuevre is initiated, in which $v^i(t)$ or $v^j(t)$ is rotated by a small angle $\psi > 0$. This rotation effectively transforms the head-on collision configuration to a simple collision configuration. Once this is done, the avoidance constraints defined in $(3.4.10)$ and

(3.4.11) solves in real time. Figure 3.10 shows the trajectory obtained for two-vehicle re-configuration with head-on collision avoidance, by applying this strategy.

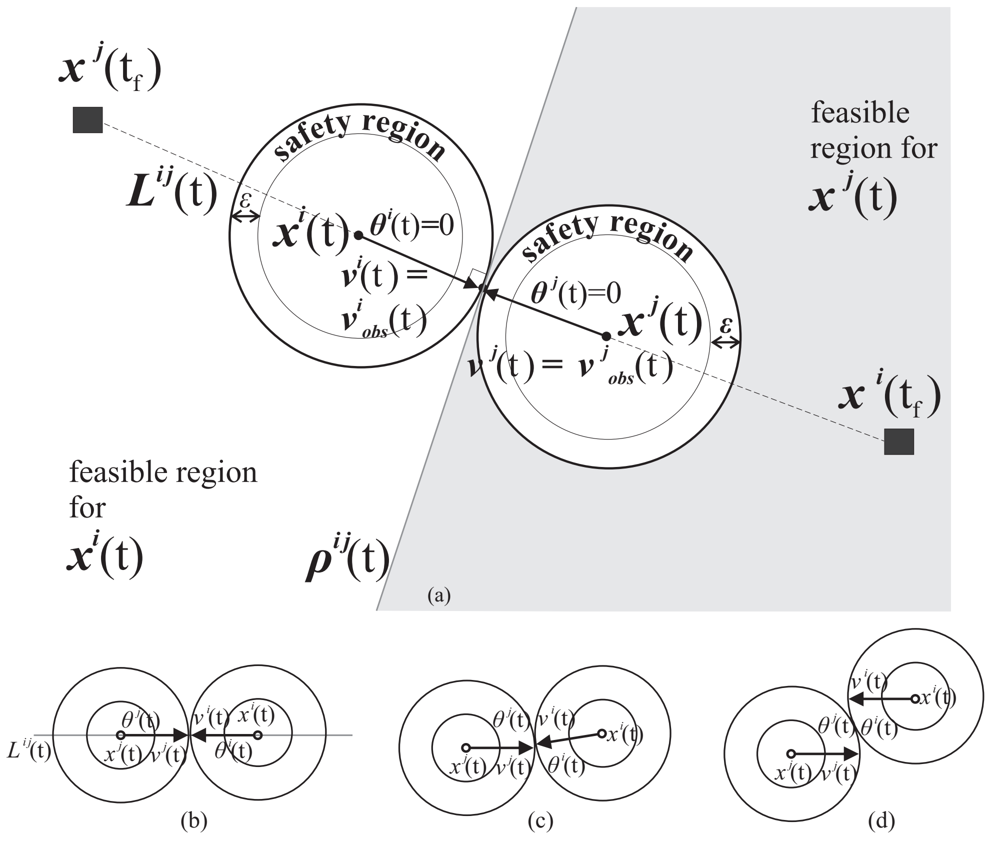


Figure 3.9: Head-on collision problem.

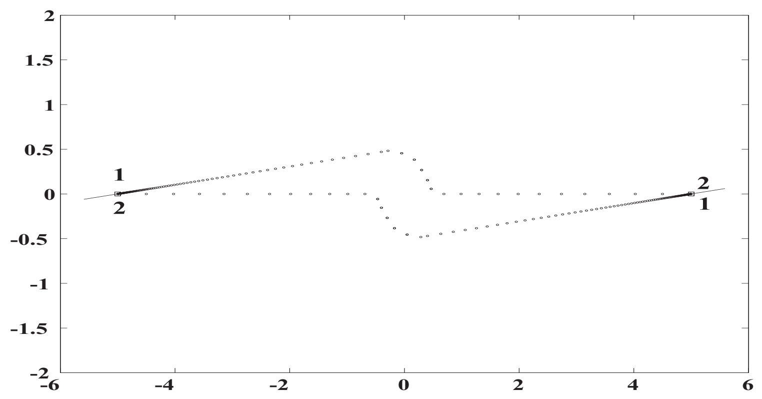


Figure 3.10: Head-on collision avoidance for two vehicles.

3.4.3.3 Detecting and Resolving Cross-Path Collision for Two Vehicles

A cross-path collision problem is any configuration in which $D^{ij}(t) \leq (r^i + r^j)$ and the current vector directions (or attitude vectors) $v^i(t)$ and $v^j(t)$ of the motion of vehicles i and j are on the same side of $L^{ij}(t)$, and the attitude vectors $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$ are not parallel. Note that since $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$ are not parallel, a point (for 2D) or line (for 3D) of intersection can be computed for both vectors. Figure 3.11 is an example of a cross-path collision problem. Note that for the avoidance process, the AC algorithm attempts to expand the angles $\theta^i(t)$ and $\theta^j(t)$ to an angle $\geq \pi/2$. Based on this initial configuration therefore, $v^i(t)$ and $v^j(t)$ will remain parallel or close to parallel, but not in opposite directions. If this continues the desired goal positions may never be reached, or may be reached after a great deal of effort.

To resolve this problem, it is required to determine whether the two vehicles are indeed in a cross-path configuration. The task is therefore to determine if there exists a point or line of intersection between $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$, and if such an intersection lies on one side of L^{ij}.

Determining Cross-Path Collision in 3D and 2D

To determine cross-path collision between i and j in 3D, two planes PL^i and PL^j are defined, both parallel to the z axes of the world coordinate frame (see Figure 3.12). Each plane must contain the vectors $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$, as shown in the figure. Therefore the plane PL^i is defined as the set $(N^i(t), x^i(t), v^i(t), z^i(t))$, where $z^i(t)$ is a point chosen above or below $x^i(t)$ or $v^i(t)$ on the z axis, and $N^i(t)$ is the normal vector perpendicular to $x^i(t), v^i(t)$ and $z^i(t)$. Once PL^j is similarly defined, the intersection of the two planes can be computed using techniques from computational geometry (Eberly, 2001; van den Bergen, 2004). If the two planes are not parallel, the computation of planes intersection will return a line l^{ij}. Once this line is determined, the next step is check if the line is on one side of the plane parallel to the z axis and containing the points $x^i(t)$ and $x^j(t)$.

An easy way to do this is to compute the perpendicular distances from the points $x^i(t)$, $v^i(t)$, $x^j(t)$ and $v^j(t)$, to l^{ij}, which can also be done using techniques from computational geometry (Eberly, 2001; van den Bergen, 2004).

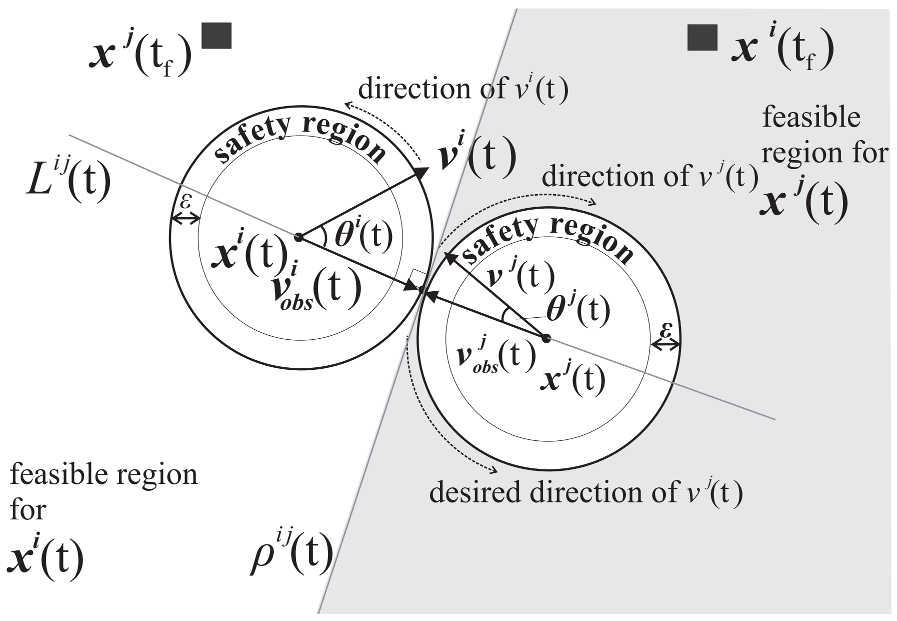


Figure 3.11: Cross-path collision trajectory.

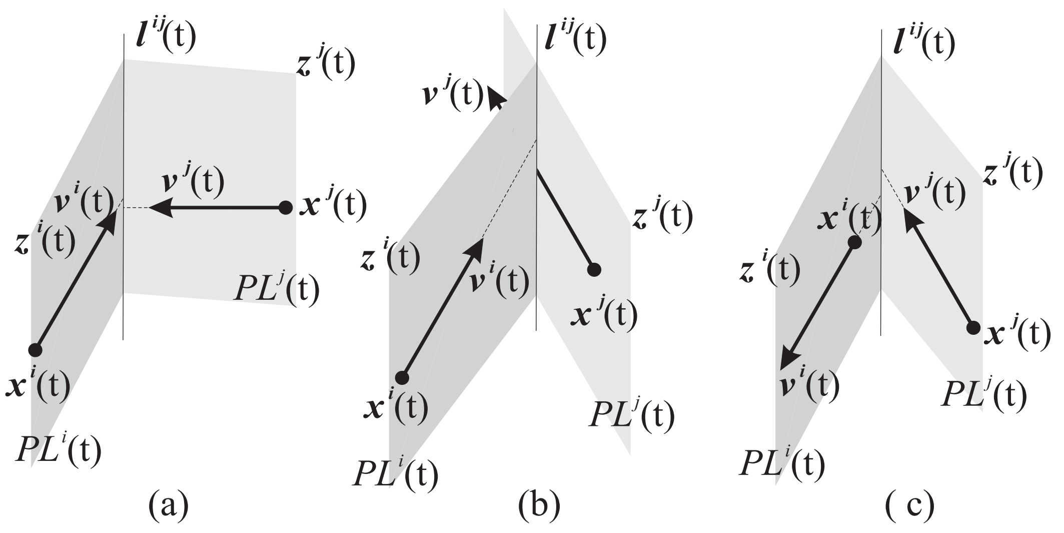


Figure 3.12: Determination of cross-path collision in 3D.

Let the corresponding distances be:

$$\begin{aligned} d_x^i(t) &= \left\| x^i(t) - l^{ij} \right\|, \\ d_v^i(t) &= \left\| v^i(t) - l^{ij} \right\|, \\ d_x^j(t) &= \left\| x^j(t) - l^{ij} \right\|, \\ d_v^j(t) &= \left\| v^j(t) - l^{ij} \right\|. \end{aligned}$$

If $d_v^i(t) \leq d_x^i(t)$ and $d_v^j(t) \leq d_x^j(t)$, then the line of intersection is in front of both vehicles, and a cross-path collision is imminent as shown in Figure 3.12 (a) and (b). Otherwise there is no cross-path conflict as shown in Figure 3.12 (c).

The analysis is simpler in the 2D case. Instead of l^{ij}, we search for a point p^{ij}, which is the point of intersection of the lines passing through $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$, as shown in Figure 3.13. If indeed such a point is found, we compute:

$$\begin{aligned} d_x^i(t) &= \left\| x^i(t) - p^{ij} \right\|, \\ d_v^i(t) &= \left\| v^i(t) - p^{ij} \right\|, \\ d_x^j(t) &= \left\| x^j(t) - p^{ij} \right\|, \\ d_v^j(t) &= \left\| v^j(t) - p^{ij} \right\|. \end{aligned}$$

See Figure 3.14 for an illustration of the computation of d. Figure 3.14 (a) is a cross-path collision configuration but (b) is a simple collision configuration. The solution strategy adopted is to convert any cross-path configuration such as (a) to a simple configuration such as (b).

To do this one only has to move either $v^i(t)$ or $v^j(t)$ to the other side of $L^{ij}(t)$ (or onto the line $L^{ij}(t)$). A simple strategy to decide which $v(t)$ should be moved in order to obtain smoother phase transition is to measure $\theta^i(t)$ and $\theta^j(t)$. If $\theta^j(t) < \theta^i(t)$ then $v^j(t)$ should be moved. This is done by swapping $v^j(t)$ and $v_{obs}^j(t)$, which immediately results in a simple collision reconfiguration. Thereafter, when the Q-CAC algorithm expands $\theta^j(t)$, it is the former $v_{obs}^j(t)$ (which is now the new $v^j(t)$) that moves, while the former $v^j(t)$ (which is now the new $v_{obs}^j(t)$) remains static.

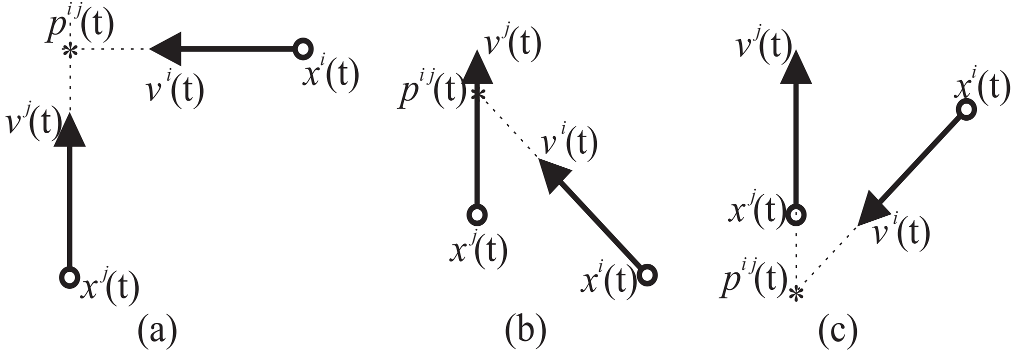


Figure 3.13: Determination of cross-path collision in 2D.

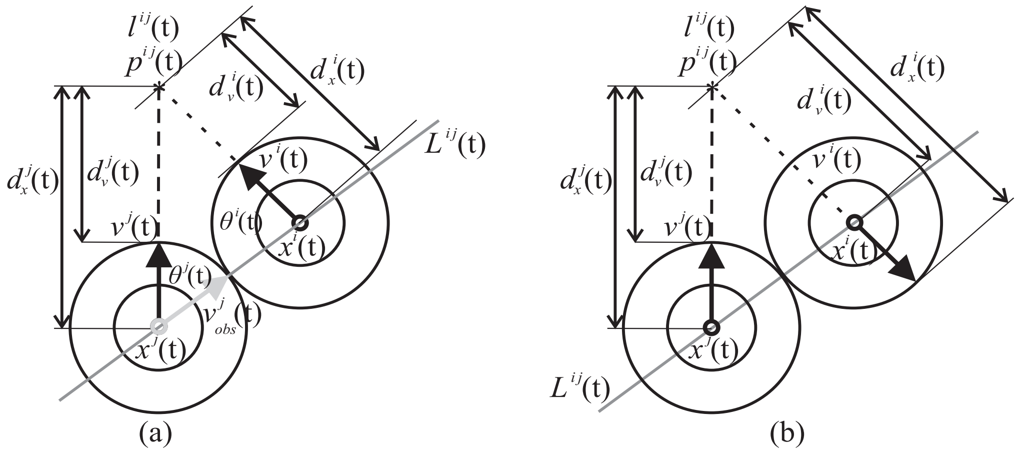


Figure 3.14: Computation of d for determination of cross-path collision.

Therefore, if a cross-path trajectory is determined, to resolve the problem it is sufficient to swap the variables in one of the avoidance constraints (3.4.10) or (3.4.11). For example, (3.4.10) may be left as it is and (3.4.11) is rewritten in the form

$$\begin{bmatrix} 2(\mu+\cos\phi) & \begin{bmatrix} v_{obs}^j(k+2) \\ v^j(k+2) \end{bmatrix}^T \\ \begin{bmatrix} v_{obs}^j(k+2) \\ v^j(k+2) \end{bmatrix} & \mathbf{M} \end{bmatrix} \geq 0.$$

The transformation of Figure 3.11 resulting from this new LMI is shown in Figure 3.15 (a).

Suppose $\theta^j(t)$ of Figure 3.14 (a) was expanded by $90°$ at time t after the swapping of

$v^j(t)$ and $v^j_{obs}(t)$, then $v^j(t)$ has changed position at time t. The new configuration for both vehicles at time t will look like the configuration in Figure 3.15 (b). After this, the new positions $x^i(t+\Delta t)$ and $x^j(t+\Delta t)$ can be computed along the current vectors $\overrightarrow{x^i(t)v^i(t)}$ and $\overrightarrow{x^j(t)v^j(t)}$. The trajectory obtained by applying this strategy to cross-path collision avoidance for two vehicles in 2D and 3D are shown in Figure 3.16.

3.4.3.4 Resolving Cross-Path Collision for More Than Two Vehicles

A more challenging situation is when more than two vehicles are involved in the cross-path conflict trajectory, as shown in Figure 3.17. In addition to our earlier assumption that each vehicle is bounded by a circle or sphere **S**, we further assume that only three vehicles can be involved in a cross-path conflict configuration at any time t. This can be seen from Figure 3.18 (provided that the radii are approximately equal). Observe from the figure that the safety region of any one of the vehicles intersects the safety regions of at most two other vehicles, and this does not change if the number of vehicles is increased. Therefore a conflict resolution strategy is developed in this section for the three vehicles cross-path configuration of Figure 3.17.

For any vehicle i, whose attitude vector $v^i(t)$ is in cross-path configuration with vehicles j and k, we are concerned only about the two bounding obstacle vectors $v^{ij}_{obs}(t)$ and $v^{ik}_{obs}(t)$. Any other vehicles intersecting the safety region of i will be ignored. The challenge of the above problem is that the Q-CAC algorithm attempts to expand $\theta^{ij}(t)$ and $\theta^{ik}(t)$ simultaneously, therefore $v^i(t)$ will remain stuck between $v^{ij}_{obs}(t)$ and $v^{ik}_{obs}(t)$ if a cross-path conflict is not resolved.

Vehicles i, j and k may be able to break out of the configuration in several ways, for example by moving backwards (if they can). Suppose there are other vehicles behind any one of the vehicles taking a reverse, then there may be possible collisions. Therefore, only positive non-zero velocities are desired, this rules out any avoidance measure that will require any of the vehicles to stop or retract. To achieve this, a *counter-clockwise* avoidance measure is defined in which for any vehicle in the three-cross-path conflict, the left bounding obstacle vector is always chosen as the cross-path obstacle vector for avoidance.

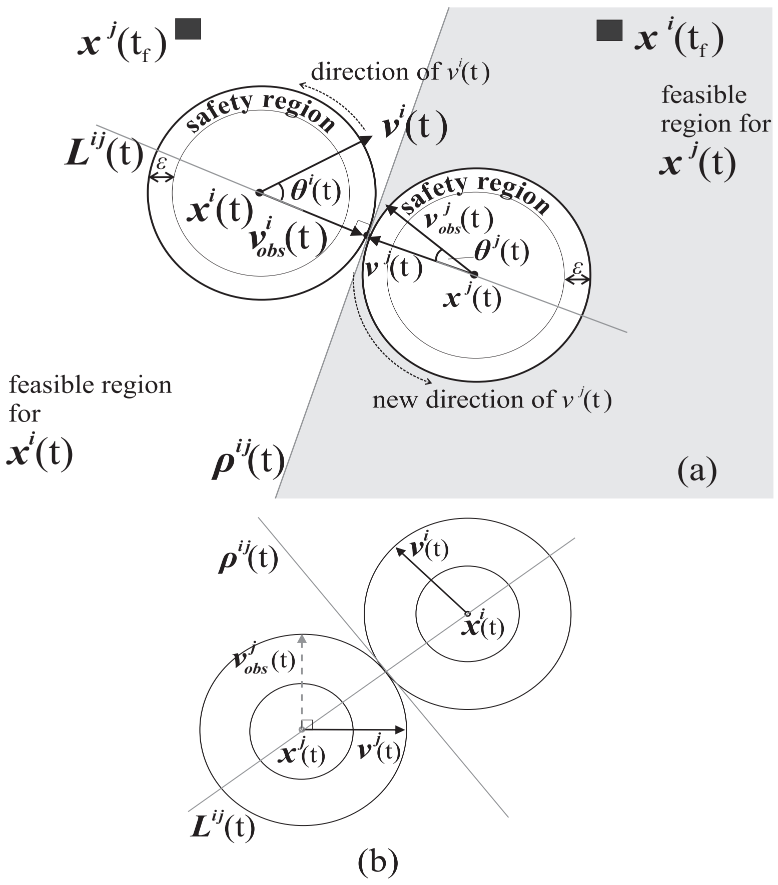


Figure 3.15: The effects of cross-path conflict resolution.

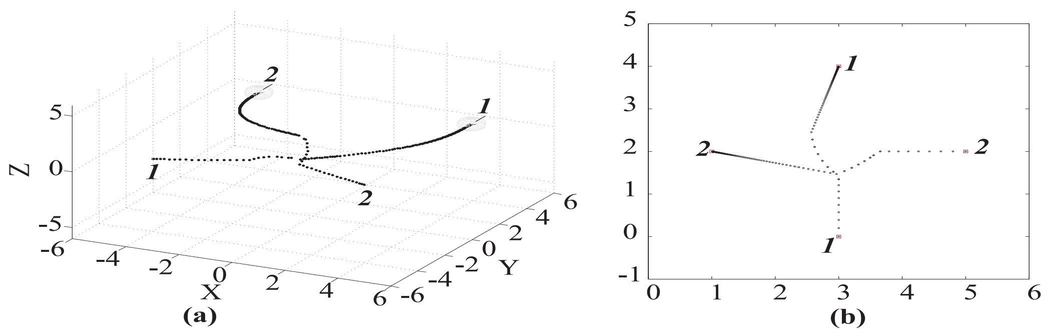


Figure 3.16: Cross-path avoidance trajectory for 3D (a) and 2D (b).

Essentially, the problem is broken down into three independent sets of cross-path conflicts and each of the pairs of vehicles (i, j), (j, k) and (k, i) are treated as being in separate independent cross-path conflicts as shown in Figure 3.19. For a counter-clockwise avoidance manoeuvre, the left vehicle is always chosen to avoid. Therefore, in Figure 3.19 (a)

$v^i(t)$ will be swapped with $v^{ij}_{obs}(t)$ keeping $v^j(t)$ constant. In Figure 3.19 (b) $v^j(t)$ will be swapped with $v^{jk}_{obs}(t)$ keeping $v^k(t)$ constant, and in (c) $v^k(t)$ will be swapped with $v^{ki}_{obs}(t)$ keeping $v^i(t)$ constant. Since this process is required to be a cooperative action among the three vehicles, the initial configurations are used in the attitude change maneuvres, and not the new attitudes resulting from earlier attitude change maneuvres. This means that after the cross-path resolution for (a), and (b), the one for (c) is done using the old value of $\overrightarrow{x^i(t)v^i(t)}$, not the new one resulting from the cross-path conflict resolution on (a).

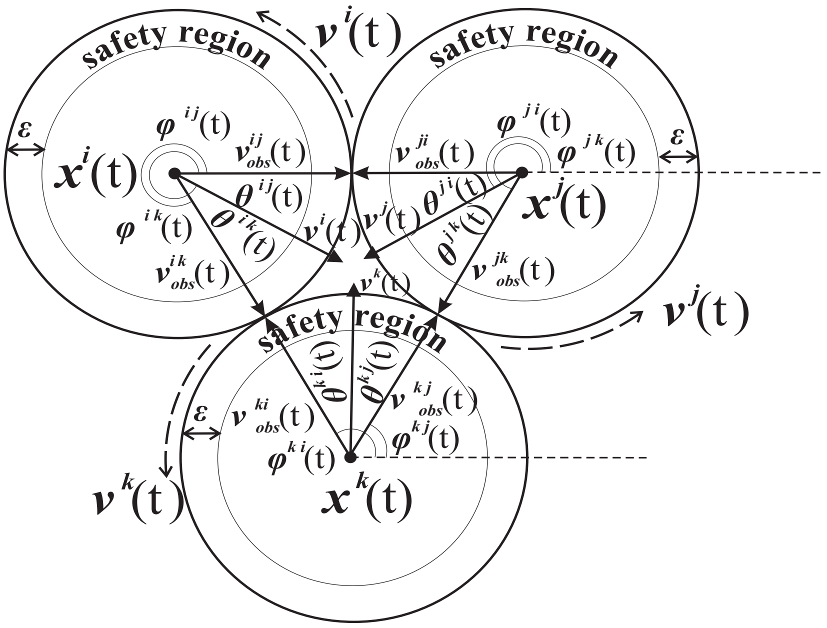


Figure 3.17: Three-vehicle cross-path trajectory problem.

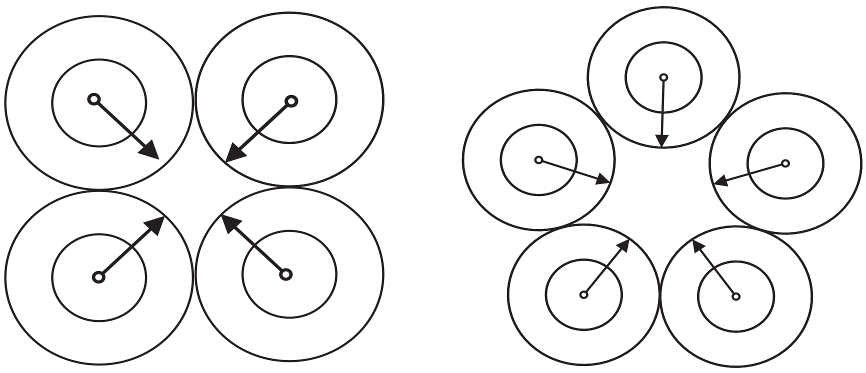


Figure 3.18: Cross-path trajectory problem for more than three vehicles.

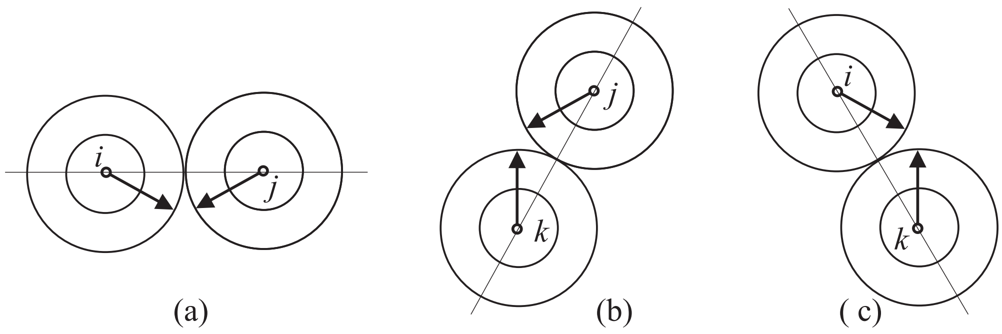


Figure 3.19: Problem isolation of the three-vehicle cross-path trajectory problem.

The remaining issue is therefore to determine which vehicle is to the left, or to the right side of another vehicle whenever a three-cross-path conflict is reported, given the current positions and attitudes of the three vehicles. This can be done by applying techniques from computational geometry. Given $x^i(t)$, $v^i(t)$, $x^j(t)$, $v^j(t)$, the following procedure will determine if j is to the left or to the right of i:

$$\begin{aligned}
D^{ij}(t) &= \left\|x^i(t) - x^j(t)\right\|, \\
\bar{D}^{ij}(t) &= \frac{D^{ij}(t)}{\|D^{ij}(t)\|}, \\
\bar{D}^{ij}_{\theta}(t) &= \tan^{-1}\left(\frac{\bar{D}^{ij}_y(t)}{\bar{D}^{ij}_x(t)}\right), \\
\theta^i(t) &= \tan^{-1}\left(\frac{v^i_y(t)}{v^i_x(t)}\right).
\end{aligned}$$

If $x^i(t)$, $v^i(t)$, $x^j(t)$, $v^j(t)$ all satisfy the inequality $\theta^i(t) \leq \bar{D}^{ij}_{\theta}(t) \leq \theta^i(t) + \pi/2$, then j is to the left of i, otherwise j is to the right of i, and vice versa.

To extended this formulation to n vehicles in a cross-path conflict, each pair of adjacent vehicles is chosen in independent cross-path conflicts and solved separately. The notion can also be further extended to vehicles of arbitrary size. Consider Figure 3.20: five independent cross-path problems can be extracted from the five-cross-path problem, and the problems can be solved separately and in any order. The cross-path sets are (i, j), (j, k), (k, l), (l, m), (m, i).

Thus, for the configuration of Figure 3.17, it is sufficient to enforce the following constraints:

$$\begin{bmatrix} 2(\mu+\cos\phi) & \begin{bmatrix} v^{ij}_{obs}(k+2) \\ v^{i}(k+2) \end{bmatrix}^T \\ \begin{bmatrix} v^{ij}_{obs}(k+2) \\ v^{i}(k+2) \end{bmatrix} & \mathbf{M} \end{bmatrix} \geq 0,$$

$$\begin{bmatrix} 2(\mu+\cos\phi) & \begin{bmatrix} v^{jk}_{obs}(k+2) \\ v^{j}(k+2) \end{bmatrix}^T \\ \begin{bmatrix} v^{jk}_{obs}(k+2) \\ v^{j}(k+2) \end{bmatrix} & \mathbf{M} \end{bmatrix} \geq 0.$$

$$\begin{bmatrix} 2(\mu+\cos\phi) & \begin{bmatrix} v^{ki}_{obs}(k+2) \\ v^{k}(k+2) \end{bmatrix}^T \\ \begin{bmatrix} v^{ki}_{obs}(k+2) \\ v^{k}(k+2) \end{bmatrix} & \mathbf{M} \end{bmatrix} \geq 0.$$

Note that the counterclockwise avoidance measure formulation presented above is more critical for the 2D navigation than for 3D, because the 2D space is more constrained since there are only two degrees of freedom. Simply put, each vehicle in 2D which is involved in collision conflict with another vehicle has only one choice of direction to turn to in the plane, and that choice should be the opposite direction to the choice taken by the other vehicle, for them not to collide. The counterclockwise measure ensures that this is so every time.

Figure 3.21 shows the results for applying the approach presented above to cross-path avoidance for three and four vehicles in 2D and 3D.

When the head-on or cross-path collision problems occur for vehicles in 3D, the counterclockwise avoidance strategy need not be strictly enforced because there are more degrees of freedom for the vehicles. In many situations for the 3D case the Q-CAC based algorithm finds feasible avoidance trajectories without the counterclockwise measure. This can be observed from the avoidance trajectories generated for the 3D cases in Figure 3.21.

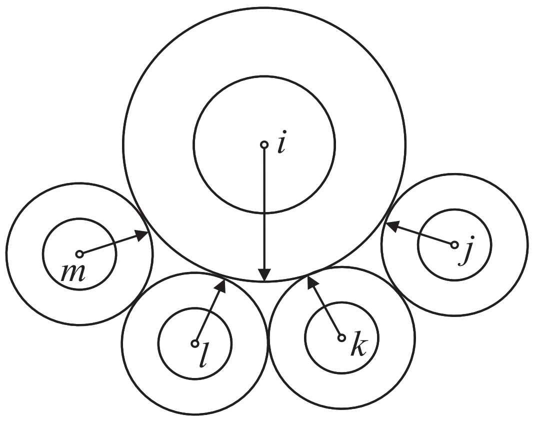


Figure 3.20: Five-vehicle cross-path trajectory problem due to unequal size.

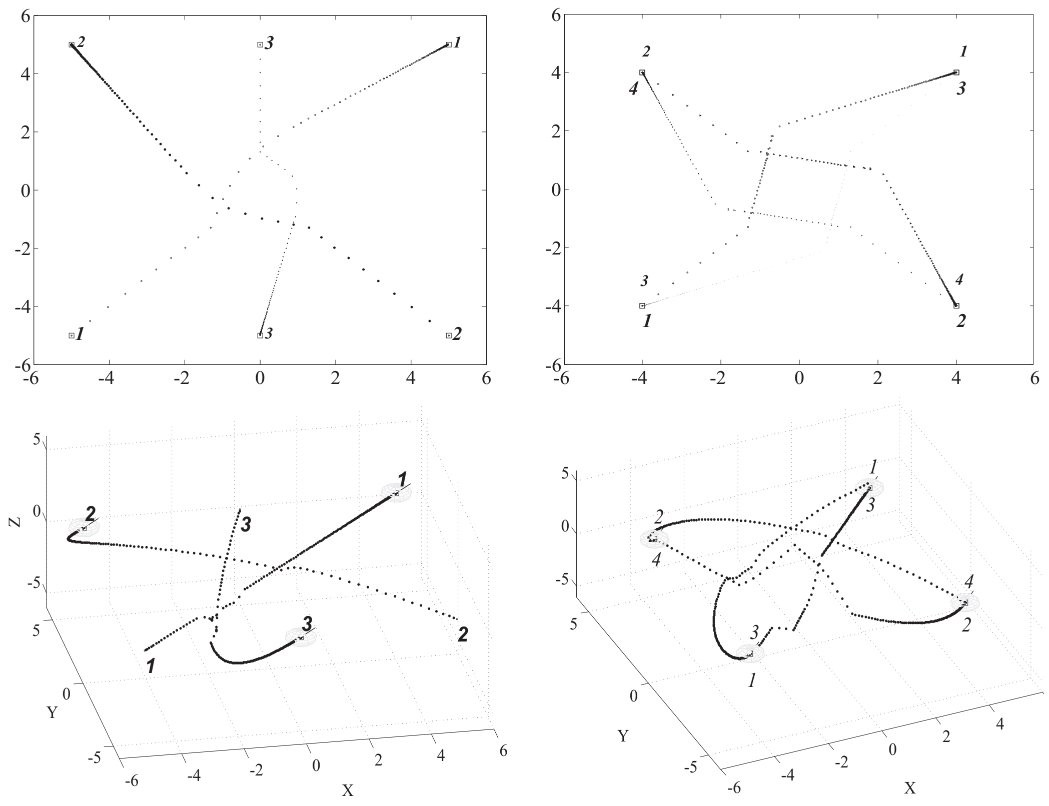


Figure 3.21: Cross-path avoidance for three and four vehicles in 2D (top) and in 3D (bottom).

However, when the vehicles in 3D are constrained to move only at the same altitude, then the counterclockwise avoidance strategy must be enforced for effective avoidance. This process is automatically enforced by the design of the algorithm and need not be done manually.

3.4.4 Consensus With Q-CAC Based Avoidance

Given (3.4.1) or (3.4.2), (3.4.5) and (3.4.9), the Q-CAC avoidance algorithm is incorporated into the consensus framework by using a two-step approach in which the consensus problem is solved simultaneously with the Q-CAC problem in every time step. Thus, for each time k, a future position of each vehicle is calculated, say up to $k + \iota$ steps, [4] by propagating (3.4.1) or (3.4.2) to obtain future consensus positions x_c^i. Among the future positions, the safe positions x_{safe}^i $(i = 1, \cdots, n)$ are identified. Then for each x^j not satisfying x_{safe}^i, the Q-CAC optimization problem stated above is solved to generate the desired safe control input u_{safe}^j. The process is summarised in Algorithm 1.

Algorithm 1 CONSENSUS-WITH-AVOIDANCE

```
Ensure: $x^i(k) \neq x_d^i \; \forall x^i(k) \in \mathbf{x}(k)$
  while $x^i(k) \neq x_d^i \; \forall x^i(k) \in \mathbf{x}(k)$ do
    $\mathbf{x}_c \leftarrow \mathbf{Ax}(k) + \Delta t\mathbf{Bu}(k)$
    if $\left\|x_c^i - x_c^j\right\| < 2r$ then
      solve (3.4.6),(3.4.11),(3.4.12)
      $u_{safe}^i \leftarrow \frac{r^*}{2} v^i(k)$
      $u_{safe}^j \leftarrow \frac{r^*}{2} v^j(k)$
    else
      $u_{safe}^i \leftarrow u^i(k)$
      $u_{safe}^j \leftarrow u^j(k)$
    end if
    $\mathbf{x}(k+1) \leftarrow \mathbf{Ax}(k) + \Delta t\mathbf{Bu}_{safe}$
  end while
```

Once a safe attitude vector $v^i(k)$ is computed at time k for any i, the next position $x^i(k+1)$ is computed as a point a distance $\frac{r^{*i}}{2}$ from the current position, along the vector $v^i(k)$. Note that $v^i(k)$ is normalized to keep the computed control bounded. Whether there are intersections of the safety regions or not, the safety of the algorithm is guaranteed by bounding the control size within the interval $0 < u^i \leq \frac{r^{*i}}{2}$.

Again, consider that the size of control computed at each time using Laplacian matrices is directly proportional to the algebraic connectivity of the graph, and inversely proportional to the magnitude of the current time k. Thus, while the early large values of **u** are

[4] ι is some positive integer which indicates the number of optimization time steps to plan ahead.

unsafe for collision avoidance (and must be bounded), the latter very small values of **u** slows down the rate of convergence. Generally, collisions are less likely to occur when the vehicles are closer to their goal positions, and this is exactly when convergence is slower. To obtain a constantly bounded **u** which guarantees collision avoidance and a high speed of convergence, the following modifications to (3.4.1) and (3.4.2) are proposed. For the leader follower architecture:

$$\begin{aligned}\mathbf{u} = &-\eta \log_{10}(k+1)\frac{\Delta t}{2\lambda_2(\mathbf{L})}\Gamma(\mathbf{x}-\mathbf{x}^{off}) \\ &-\beta \log_{10}(k+1)\frac{\Delta t}{2\lambda_2(\mathbf{L})}\mathbf{K}(\mathbf{x}-\mathbf{x}^{off}).\end{aligned} \tag{3.4.12}$$

The corresponding protocol for the leaderless architecture is:

$$\begin{aligned}\mathbf{u} = &-\eta \log_{10}(k+1)\frac{\Delta t}{2\lambda_2(\mathbf{L})}\Gamma(\mathbf{x}-\mathbf{x}^{off}) \\ &-\beta \log_{10}(k+1)\frac{\Delta t}{2\lambda_2(\mathbf{L})}\mathbf{K}(\mathbf{x}-\mathbf{x}_d),\end{aligned} \tag{3.4.13}$$

where $\lambda_2(\mathbf{L})$ denotes the second smallest eigenvalue of $\mathbf{L}$, η is a scaling parameter for the consensus term, and β is a scaling parameter for the proportional term in (3.4.12) and (3.4.13), respectively. The logarithmic term $\log_{10}(k+1)$ and the term $\frac{\Delta t}{2\lambda_2(\mathbf{L})}$ attempt to reduce $||\mathbf{u}||$ when k is small, and increase $||\mathbf{u}||$ when k is large. The choice of η and β depends on the radius of **S**, and $\mathcal{E}$ for each vehicle. An alternative solution is to compute an unbounded **u** using (3.4.1) or (3.4.2), and for each $u^i > \frac{r^{*i}}{2}$, normalize u^i and set $u^i = \frac{r^{*i}}{2}u^i$.

3.4.5 Procedure for Implementing the Algorithm

Step-by-step details of how to implement the algorithm are presented in this section, in order to enhance the usability of the concepts developed in this chapter for motion control of autonomous vehicles. First, the basic software modules that will be required to implement the algorithm are discussed, then procedures for combining the modules to realize a centralized or decentralized implementation are outlined in a flowchart.

3.4.5.1 Required Modules

(i). *A collision detection module*, which is essentially a Euclidean distance measurement process for all the vehicles. A collision conflict is detected whenever $D^{ij} \leq (r^i + r^j)$ for any (i, j) pair, $i \neq j$. Two implementations of this module are possible: (i) a centralized collision detection module which accepts the current positions of all the vehicles and returns the positions and attitudes of all the vehicles that are in a collision configuration; (ii) a decentralized collision detection module, which accepts the current positions of all the vehicles and returns the positions and attitudes of all vehicles that are in a collision configuration with a particular vehicle.

(ii). *An attitude intersection module*, which is used to classify the type of collision configuration between any sets of vehicles involved in a collision conflict. Essentially the algorithm will sort the collision list in an ascending or descending order, then for each vehicle, it checks whether the vehicle is involved in a simple collision configuration, a head-on collision configuration, or a cross-path collision configuration with one or more vehicles. This module accepts the list of vehicles in collision configuration returned from the collision detection module. The list contains the position and attitude information for all the colliding vehicles. The output is the set of groups of vehicles involved in a collision, each labelled with of the type of collision configuration.

(iii). *A collision avoidance module*, which implements the Q-CAC algorithm that responds to the appropriate type of collision, based on the type of collision data returned for each vehicle by the attitude intersection module. The module accepts the position and attitude data for any two vehicles i and j involved in a collision configuration, and a list of data describing the type of collision they are involved in. It then computes v^i_{obs} and v^j_{obs}, which are essentially extracted from the mid point of the line connecting x^i and x^j. If vehicle i is the candidate vehicle in cross-path collision with j, then the swapping of v^i and v^i_{obs} is performed. The module will produce responses for simple, head-on or cross-path collision. Sometimes the responses are different for the three configurations, however if the module is implemented correctly, a collision free avoidance trajectory is returned. To implement this module using the LMI constraints, an LMI parser and optimization software will generally be required. For the Matlab simulated experiments presented in this chapter, we used the LMI parser Yalmip (Lofberg, 2004), and the optimization software Sedumi (Sturm, 1998).

For 2D navigation, the attitude constraints can actually be solved manually without having to solve an LMI using optimization software, because the dimension is small, which means that we could implement the Q-CAC constraints in 2D manually without using LMI, and indeed the algorithm will run faster. However, this will require additional work of having to decide whether the vehicles in a collision conflict are on the left or right side of each other, in order to determine which angles to expand. For 3D navigation, it is required to solve the LMI constraints using an optimization software because the dimension is higher.

(iv). *A direction determinant module*, which helps to decide whether the vehicles in a collision conflict are on the left or right side of each other.

Once the required modules are in place, there are two ways in which the algorithm can be implemented: centralized; and decentralized or distributed.

3.4.5.2 Implementation Procedure

Decentralized Implementation

For the decentralized implementation, each vehicle uses its own onboard sensors, or data communicated to it, to obtain the positions and orientations of other vehicles in close proximity with it. This information is used to compute the next feasible trajectory at every time step. This is essentially a single vehicle response to its goal(s) and its environment. A typical example of such a setup is the RoboCup medium sized or humanoid league.

The flowchart in Figure 3.22 is an example of a decentralized implementation for control of vehicle i. Next, a step-by-step explanation of the flowchart is present.

Step 1. Read current data $\mathbf{x}(t), \mathbf{x}_d, \mathbf{v}(t), \mathbf{r}^*, \mathcal{E}$.

Step 2. Check if any vehicle has not reached its desired position. This is done by computing $D_d(t) = \|\mathbf{x}_d - \mathbf{x}(t)\|$. If $D_d(t) > \epsilon$ for some small $\epsilon > 0$, proceed to Step 3, otherwise stop, because all vehicles have reached their goals.

Step 3. Detect collisions between vehicle i and all other vehicles by computing $D^{ij}(t) = \|x^i(t) - x^j(t)\| \ \forall\, j, j \neq i$. A range sensor on vehicle i can be used instead to do this. If $D^{ij}(t) \leq (r^i + r^j)$ for any j then return a list of vehicles that violate the safety region of i and proceed to Step 4, otherwise proceed to Step 8.

Step 4. For the list of colliding vehicles compute the points of intersection. In the flowchart a function $intrsct^{ij}(t) = lineintrsct(x^i(t), v^i(t), x^j(t), v^j(t))$ is implemented. For 2D the value returned is $p^{ij}(t) = intrsct^{ij}(t)$, and for 3D the value returned is $l^{ij}(t) = intrsct^{ij}(t)$. If $intrsct^{ij}(t)$ contains real numbers proceed to Step 5, otherwise $intrsct^{ij}(t) = \infty$ and there is no intersection. Then set two intersection determinant parameters $infront^i(t) = 0$, $infront^j(t) = 0$ and proceed to Step 6.

Step 5. For vehicles where $p^{ij}(t)$ or $l^{ij(t)}$ exists, compute the intersection determinant parameters $[infront^i(t), infront^j(t)] = attintrsct(x^i(t), v^i(t), x^j(t), v^j(t))$ and proceed to Step 6.

Step 6. Compute an avoidance vector for i. The function $[v^i(t), v^j(t)] = Q\_CAC()$ uses $infront^i(t), infront^j(t)$ to determine whether there is a simple, head-on or cross-path collision between vehicles i and j. It can be implemented to return a new value for $v^i(t)$ and retain the old value of $v^j(t)$, or to compute new values for both $v^i(t)$ and $v^j(t)$. However, it is sufficient to compute a new value for $v^i(t)$ while keeping $v^j(t)$ constant. For centralized control the process just has to be repeated for all vehicles in a collision configuration.

Step 7. Compute a bounded control based on the new unit avoidance vector $v^i(t)$, $u^i(t) = \frac{r^{*i}}{2} v^i(t)$, and proceed to Step 10.

Step 8. There is no collision. Compute a proportional or a consensus based control. Proportional control $u^i(t) = x^i_d - x^i(t)$. Consensus control $u^i(t) = -\mathbf{L}(x^i(t) - (x^i)^{off})$

for decentralized control. Equations (3.4.12) or (3.4.13) are used for computing centralized consensus control. After computation, normalize the control $\bar{u}^i(t) = \frac{u^i(t)}{\|u^i(t)\|}$ and test if $\bar{u}^i(t) > \frac{r*^i}{2}$. If $\bar{u}^i(t) > \frac{r*^i}{2}$ proceed to Step 9, otherwise proceed to the Step 10.

Step 9. Compute a bounded control $u^i(t) = \frac{r*^i}{2}\bar{u}^i(t)$.

Step 10. Update the position of i, $x^i(t + \Delta t) = x^i(t) + u^i(t)$. Note that this update step is only necessary for simulation, but for real implementation the computed control $u^i(t)$ is sent to the motion actuators of vehicle i to control its motion.

Centralized Implementation

For the centralized implementation a central controller uses external sensors to obtain the current positions and headings of all the vehicles, and then uses this information to compute the next feasible trajectories for all the vehicles. The newly computed collision free trajectories are converted to control inputs which are sent to control the vehicles. A typical example of such a setup is a central computer controlling a team of soccer robots using an overhead camera to obtain the current robot positions and attitudes, e.g. the RoboCup small sized league. Essentially, the centralized controller performs the same tasks listed in the flowchart in Figure 3.22. However, the controller stores a record of the current state after Step 1. The stored record then is used to perform Steps 2 to 9 for all vehicles (no updated v^i is used). After control $\mathbf{u}(t)$ has been computed for all vehicles, Step 10 is performed simultaneously for all vehicles. For simulation $\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \mathbf{u}(t)$. For actual implementation $\mathbf{u}(t)$ is sent to control the motion of the real vehicles.

3.5 Simulation Results

In this section, five simulated experiments are presented to demonstrate Algorithm 1[5], using protocol (3.4.12) with Topology 1. In the first three experiments a group of soccer robots will reconfigure to desired formations while avoiding each other along the way.

[5]With $\iota = 1$.

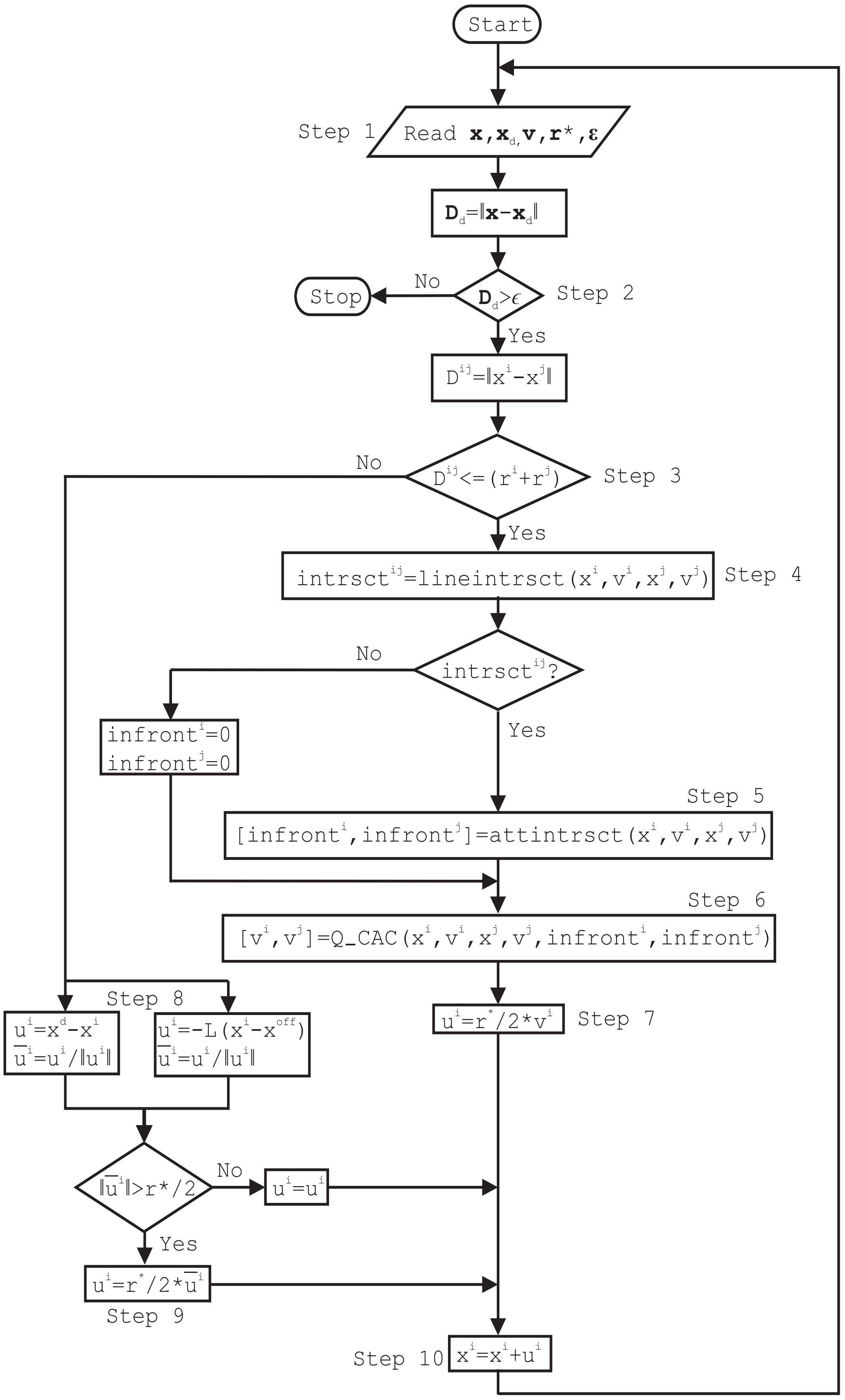


Figure 3.22: Decentralized implementation.

Note that in the experiments the robots' final positions were deliberately arranged to make the reconfiguration process difficult so as to observe the collision avoidance capability of the algorithm. The RoboCup small sized league (SSL) specification was chosen to define parameters, however eleven team members are used. The members are labelled $1, \cdots, 11$ using the smaller fonts in the figures, where 1, the goal keeper, is the leader robot. The larger fonts $1, \cdots, 11$ are the initial positions of the robots, while the asterisk $(*)$ mark the desired final positions. The robots are homogeneous, and for each robot $r^* = 85mm$, and $\mathcal{E} = 90mm$, while the dimension of the SSL soccer pitch is $6050mm \times 4050mm$. For the three experiments $\eta = 0.5, \beta = 0.5$, and the same initial configuration was used.

In the fourth experiment, the UAV reconfiguration problem initially presented in Figure 3.3 is repeated here. The fifth experiment is to test the capability of the algorithm for collision avoidance in 2D and 3D cluttered spaces using a simplified setup consisting of two vehicles. All of the simulations were done with Matlab R2009a on an Intel(R) Core(TM)2 Duo P8600 @ 2.40GHz with 2 GB RAM, running Windows 7.

3.5.1 Consensus with Q-CAC Based Reconfiguration to Soccer Play-Off Formation

In this experiment, the robots will reconfigure to a soccer *play-off* formation with collision avoidance. The initial configuration is an irregular pattern, indicated in Figure 3.23. The multi-path planning problem took 244 time steps to solve, resulting in a total computation time of $7s$, in which 203 avoidance attempts were made, and there were no collisions.

The difficulty of the path planning problem in this experiment can be observed from the fact that each vehicle except for vehicles 1, 2 and 11, was crossing the pitch to a goal position that was closer to the initial position of another vehicle. This strategy resulted in more conflict problems solve. A simpler strategy would employ a nearest distance matching algorithm to pair each vehicle to a goal position that is closer to its initial position, provided there are no permanently assigned positions (except for the goal keeper position). This way, the number of conflicts can be reduced.

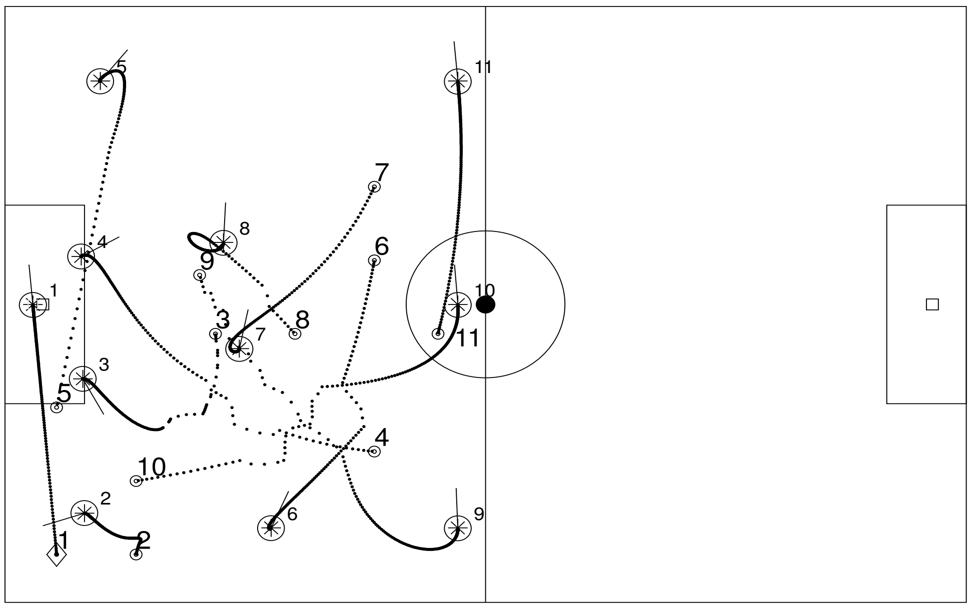


Figure 3.23: Consensus based collision free reconfiguration of eleven robots to a soccer *play-off* formation using Topology 1.

3.5.2 Consensus with Q-CAC Based Reconfiguration to Soccer Free-Kick-Defense Formation

In this experiment, the robots will reconfigure to a soccer *free-kick-defense* formation with collision avoidance, and the previous initial configuration was used. The multi-path planning problem took 242 time steps to solve, resulting in a total computation time of $12s$, in which 747 avoidance attempts were made, and there were no collisions. The results are in Figure 3.24.

The difficulty of the path planning problem in this experiment can be observed from the fact that each vehicle except for vehicles 1, 9, 10 and 11, was crossing the pitch to a goal position that was closer to the initial position of another vehicle.

3.5.3 Consensus with Q-CAC Based Reconfiguration to Soccer Corner-Kick-Defense Formation

In this experiment, the robots will reconfigure to a soccer *corner-kick-defense* formation with collision avoidance, and the previous initial configuration was used. The multi-path planning problem took 231 time steps to solve, resulting in a total computation time of $15s$, in which 666 avoidance attempts were made, and there were no collisions. The results are in Figure 3.25.

The difficulty of the path planning problem in this experiment can be observed from the fact that each vehicle except for vehicles 1, 8, 9, 10 and 11, was crossing the pitch to a goal position that was closer to the initial position of another vehicle.

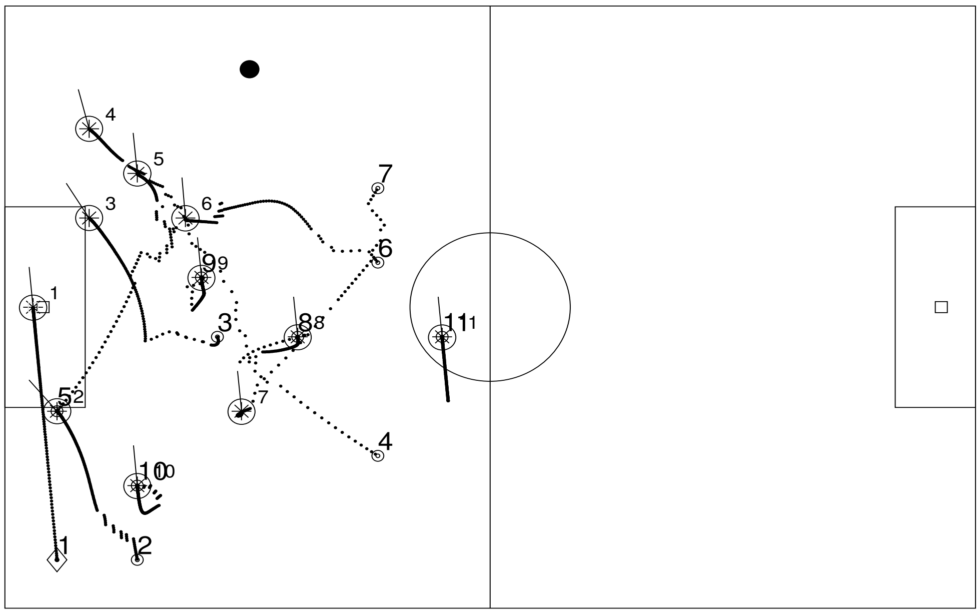


Figure 3.24: Consensus based collision free reconfiguration of eleven robots to a soccer *free-kick-defense* formation using Topology 1.

3.5.4 Consensus with Q-CAC Based Reconfigurations of Multiple UAVs

For this experiment the coordinate frame of $REGION_1$ is assumed static at $\mathcal{F}_{REGION_1} = [-4\ -4\ -4]^T$, the coordinate frame of $REGION_2$ is $\mathcal{F}_{REGION_2} = [3.5\ 3.5\ 4]^T$, and the coordinate frame of $REGION_3$ is $\mathcal{F}_{REGION_3} = [-4.1\ -3.26\ 3.2]^T$. The regions are represented as spheres. In the team, the small black sphere on the black trajectory is the leader UAV. The UAVs are homogeneous and $r^* = 0.045$ units while $\mathcal{E} = 0.09$ units, for each UAV. Trajectory data generated from Matlab was used to produce the result shown in Figure 3.26. This path planning problem took $4.165s$ to solve, and there were no collisions.

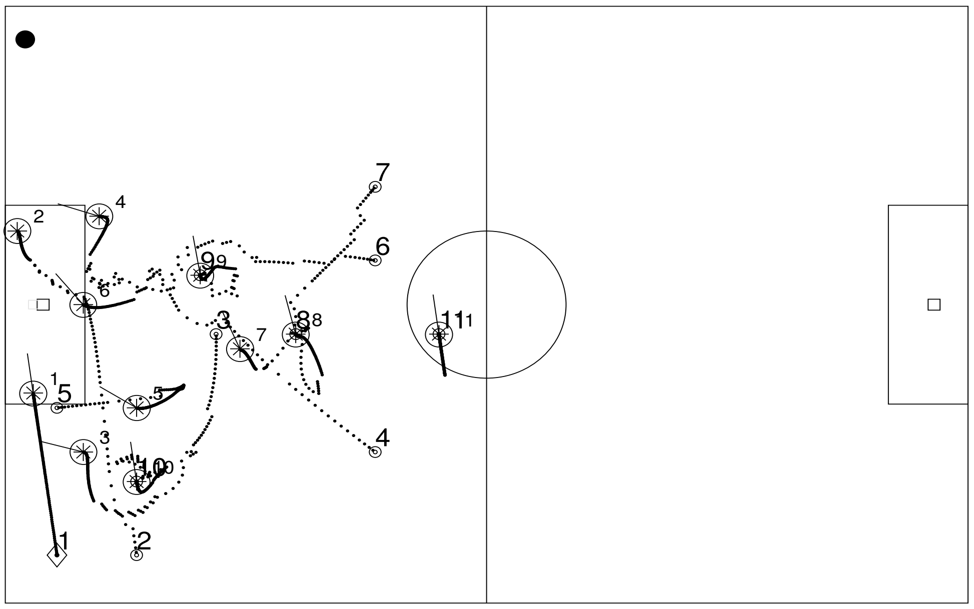


Figure 3.25: Consensus based collision free reconfiguration of eleven robots to a soccer *corner-kick-defense* formation using Topology 1.

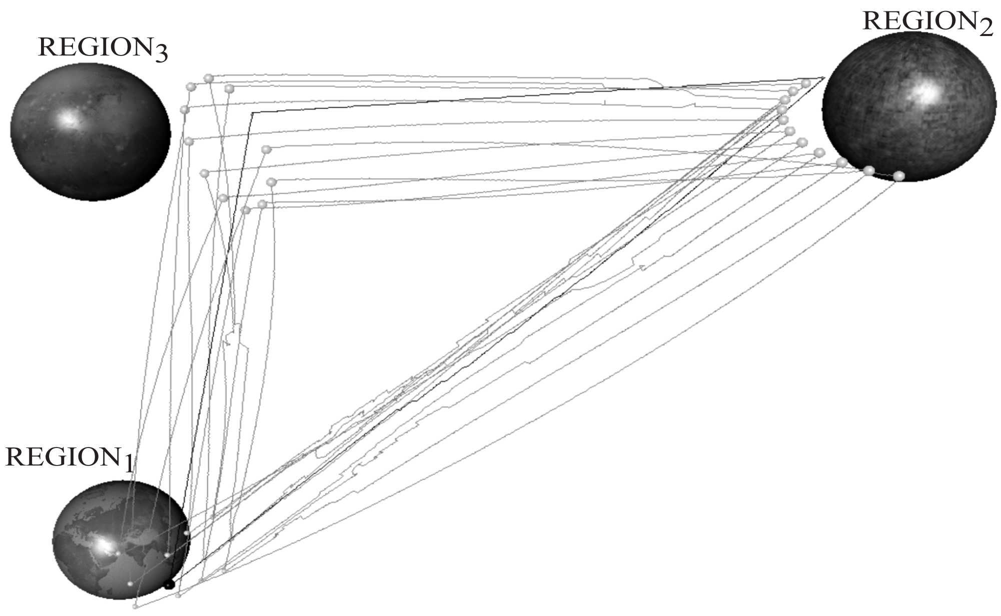


Figure 3.26: Consensus based collision free reconfigurations of twelve UAVs using Topology 1.

The difficulty of the path planning problem in this experiment can be observed from the fact that the leader UAV was engaging in lane change manoeuvres in the three laps of the entire manoeuvre. Moreover, reconfigurations to arbitrary manoeuvres is generally a difficult problem to solve.

3.5.5 Q-CAC Based Path Planning in Cluttered Space

Note that consensus algorithms are essentially for motion control in free space, therefore the multi-path algorithm developed in this chapter is basically for path planning with collision avoidance in free space. However, we needed to test the algorithm in cluttered space to observe the behaviour. Simulation was performed for a single vehicle path planning. To implement this using the algorithm, a *virtual* leader vehicle is created with a *real* follower vehicle in a two-vertex digraph configuration, such that the virtual leader vehicle never moves. The position of the virtual leader vehicle is the goal position, and the real vehicle finds a path to the static virtual leader vehicle. The result obtained for a simple cluttered 2D space consisting of polygonal obstacles, is shown in Figure 3.27. The vehicle is able to find a feasible path from the initial position $x(t_0)$ to the final position $x(t_f)$. A rough path is generated which will require trajectory smoothing to obtain a better trajectory. For the 3D experiment, the cluttered 3D space consists of several high walls with small windows as shown in Figure 3.28. Clearly, as shown in the figure, the algorithm was unable to find a complete path to the goal for this challenging cluttered 3D space. The reason why the algorithm was unable to find a complete path is discussed in the next section.

3.6 Concluding Remarks

The limitation of the Q-CAC based collision avoidance algorithm at this stage of the design is its application to multi-path planning in cluttered spaces, and this limitation is common to algorithms in which control forces generated are always strongly biased towards the goal positions. This includes most of the previous works reviewed in this chapter. The only algorithms that guarantee feasible paths in challenging cluttered 3D spaces are the probabilistic roadmap based approaches.

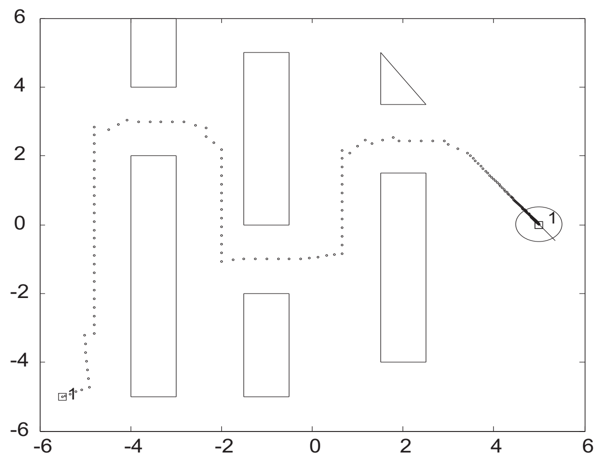


Figure 3.27: Q-CAC based path planning in 2D cluttered environment.

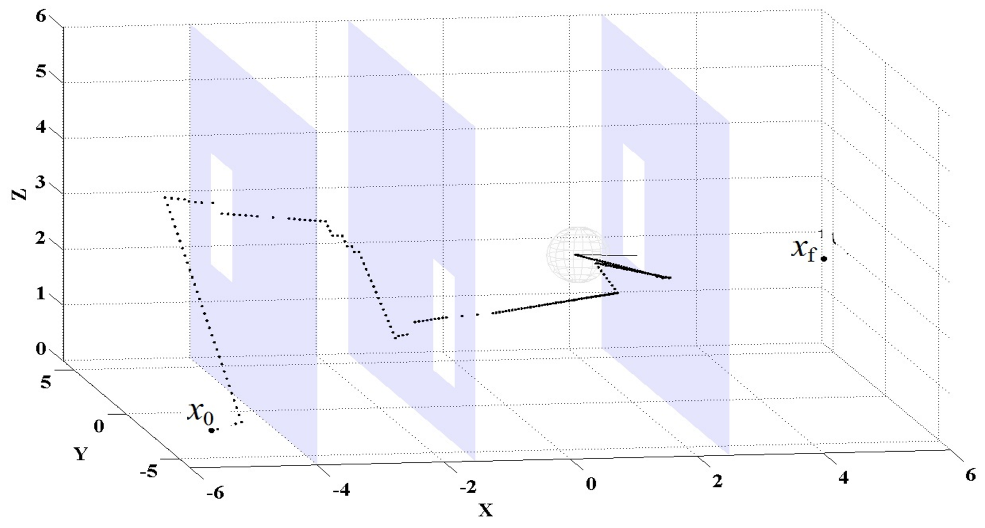


Figure 3.28: Q-CAC based path planning in 3D cluttered environment.

Considering the case for control of aerial or space vehicles, one may question the need for considering cluttered spaces in the air, or in space. However consider an air combat situation, or spacecraft navigating between fast moving asteroids, where the basic free-space collision avoidance algorithms become limited. Clearly, for some challenging clutter spaces, one *must* inevitably resort to sampling based probabilistic road map approaches such as the rapidly exploring random trees (RRT) (Kuffner and LaValle, 2000), which are best suited for path planning in cluttered spaces. Such algorithms can be used as an input to a consensus based path planning algorithm. As shown in Figure 3.29, the basic RRT algorithm took $1.7s$ to obtain a rough path to the goal, and $0.3s$ for smoothing, for the same problem presented in Figure 3.28. From this, a flyable path still has to be generated based on the motion constraints, kinematics and dynamics of the specific vehicle.

A future direction of research is to include probabilistic roadmap approaches such as RRT in the consensus with Q-CAC framework. While the RRT seems currently to be the popular choice for many applications because of its ability to find feasible trajectories in difficult cluttered spaces, and its fast speed of execution, the number of vehicles that can be allowed which can also guarantee effective collision avoidance and high speed of convergence, remains to be proved. When it comes to high speed of execution and maximization of the number of vehicles, the MILP based optimization algorithms do not scale well with the non-optimization approaches, because of computational complexity. Combining the RRT with Q-CAC, and with consensus based multi-path planning, may be potentially useful for solving cooperative navigation problems in 3D cluttered spaces. Examples include navigating around human habited environments, the environment of a space station, around space debris, or among a team of combat aircraft avoiding a hail of enemy bullets or missiles.

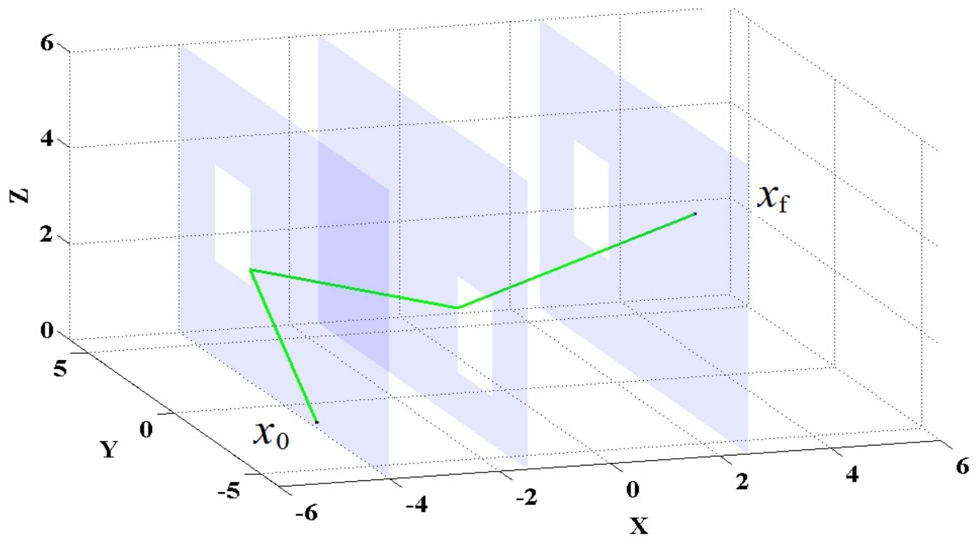


Figure 3.29: RRT based path planning in 3D cluttered environment.

Chapter 4

Constrained Attitude Control of Multiple Rigid Bodies

In this chapter, consensus theory and optimization theory are combined for distributed attitude control of a team of communicating vehicles (rigid bodies). By using the Laplacian matrix of their communication graph and a semidefinite program, a synthesis of a time varying optimal stochastic matrix **P** is done. This matrix **P** is used to drive given initial attitudes of the bodies to a consensus attitude. The concept of quadratically constrained attitude control (Q-CAC), earlier introduced, is then employed to generate an optimal collision-free attitude trajectory along the consensus trajectory. This solution technique is used to simulate coordinated rendezvous and formation acquisition of multiple rigid bodies, using a simplified model of spacecraft, in the presence of static and dynamic obstacles. A Lyapunov-based stability analysis, together with the simulation results, are also provided to demonstrate the effectiveness of the approach. Spacecraft attitude control is used as the basis for the development in this work, and the same approach can be applied to other rotating rigid bodies with fixed base (Okoloko and Kim, 2010, 2012*a*,*b*).

4.1 Introduction

Attitude control (AC) is of fundamental importance in the navigation of aircraft, spacecraft, sattelites and robots, and it has been considered extensively (Wen and Kreutz-Delgado, 1991; Kim and Mesbahi, 2004; Kim *et al.*, 2010). This control problem becomes more challenging when it involves multiple vehicles subject to various constraints in dynamic environments. In particular, when the vehicles are networked and share certain common objectives, consensus theory based on graph Laplacian matrices can be applied to control design (Fax, 2002; Ren *et al.*, 2005).

There are significant technological benefits to be derived by undertaking this research on the attitude constrained consensus problem. First, it is motivated by its potential practical applications in aerospace systems, to the development of intelligent spacecraft autonomy. Various ongoing and future space missions (e.g. Blackwood *et al.*, 2002; Unwin and Beichman, 2003) require the cooperative navigation and attitude slewing of multiple spacecraft or satellites, for such purposes as interferometry and optimal sensor coverage (see Figure 4.1). Another potential application is in cooperative robot manipulation, in which several robotic manipulators are required to jointly perform a task in unity in a constrained space, and the failure of one of the team members may lead to the failure of the team. A typical example of cooperating joints is a hexapod robot, and a more challenging example of cooperative manipulation is the *piano movers* problem (Schwartz and Sharir, 1983), in which several robots with manipulators cooperate to carry a piano from one position to another without dropping it along the way (see Figure 4.2). Other examples can be found in the work of LaValle (2006).

In this work, a rigid body will be represented by a simplified model of a spacecraft.

Attitude dynamics can be represented by unit quaternions. However, due to the nonlinearity of quaternion kinematics, it is difficult to directly apply Laplacian-like dynamics to multi-vehicle attitude control using quaternions.

In the work of Kim and Mesbahi (2004), the attitude control problem is formulated as a quadratically constrained attitude control (Q-CAC) problem, and the theories of linear matrix inequalities (LMIs) and semidefinite programming (SDP) were applied to solve it.

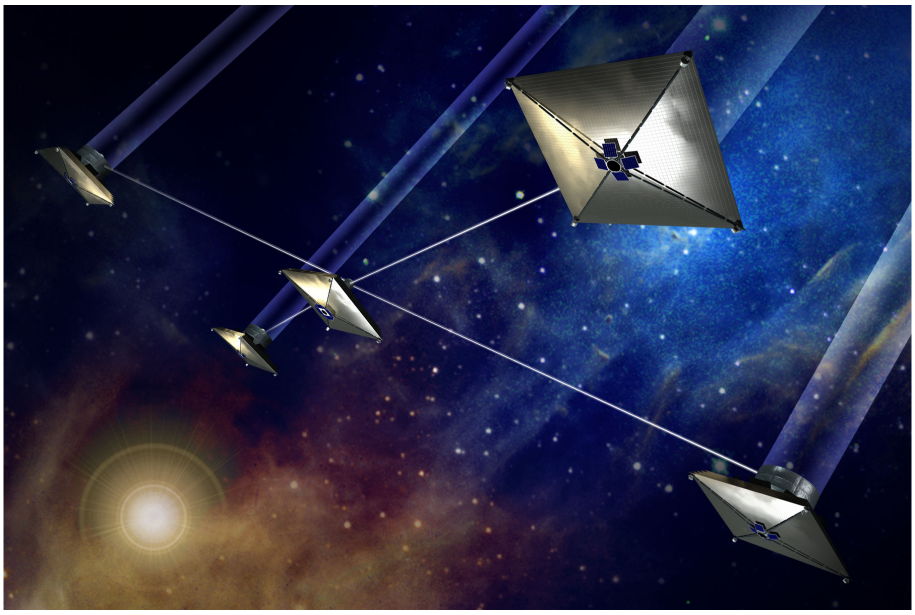


Figure 4.1: NASA's terrestrial planet finder interferometer (Unwin and Beichman, 2003).

Figure 4.2: The Piano Movers problem (Schwartz and Sharir, 1983).

They demonstrated a single spacecraft attitude re-orientation in the presence of a single static obstacle, which was further extended to two spacecraft (Kim *et al.*, 2010). In the work of Bullo *et al.* (1995), reduced spacecraft attitude stabilization on a sphere was considered, and control torques required for effective attitude stabilization were reduced from three to two. In the related work of Ren (2006), consensus protocols were applied in distributed attitude alignment of a team of communicating spacecraft flying in formation. Similarity between this approach and our approach is detailed in our previous work (Okoloko and Kim, 2010). Also, in the work of Dimarogonas *et al.* (2009), the Laplacian method was employed in leader-follower attitude control of spacecraft, using the modified Rodriquez parameters (MRP).

However, none of these aforementioned works, except the work of Okoloko and Kim (2010), apply consensus theory directly to quaternions, and none of the other works except the works of Kim and Mesbahi (2004), Kim *et al.* (2010), and Okoloko and Kim (2010), tackle the important problem of attitude cone avoidance constraints. Moreover, the works of Kim and Mesbahi (2004), Kim *et al.* (2010) and Okoloko and Kim (2010), were developed for spacecraft in the same coordinate frame, which is not a practicable situation. In our previous work (Okoloko and Kim, 2010), we first applied quaternions directly with consensus, and employed the avoidance strategy developed in the work of Kim *et al.* (2010), but the avoidance strategy was impracticable for constrained attitude control of multiple spacecraft.

As described by Kim and Mesbahi (2004), to handle the difficulty of nonlinearity in quaternion kinematics, one can cast the Q-CAC problem as a semidefinite program, subject to convex quadratic constraints. In our work (Okoloko and Kim, 2010), SDP enabled us to *synthesize* a series of Laplacian-like matrices that satisfy the constraints and achieve consensus, using optimization software tools such as Sedumi (Sturm, 1998) and Yalmip (Lofberg, 2004). However, due to computational complexity, it was difficult to implement avoidance constraints directly in the consensus framework therein. This justifies the development of a new approach in this present work. Thus, this approach extends the works of Kim and Mesbahi (2004) and Okoloko and Kim (2010), by separating the problem into two submodules. First is the synthesis of the Laplacian-like matrix, that guarantees an average consensus trajectory. Thereafter the Q-CAC problem is solved along this reference trajectory. We avoid the avoidance strategy developed by Kim *et al.* (2010) and employed by Okoloko and Kim (2010), because this is the source of the complexity. Furthermore, the solution is extended to a realistic scenario of spacecraft in different coordinate frames.

The contributions of the present work can be summarized as follows: (1) consensus theory is extended to attitude control of multiple spacecraft by the introduction of a quaternion consensus protocol; (2) the cone avoidance strategy, previously developed for a single static obstacle avoidance by Kim and Mesbahi (2004), is extended to multiple dynamic obstacles avoidance by decentralization, and this is incorporated into the consensus framework; (3) mathematical convergence analysis is provided for the quaternion-based consensus-with-avoidance framework; (4) a decentralization of the multiple spacecraft attitude control problem is developed, resulting in a new algorithm which reduces computational complexity and has a faster speed of convergence than the approach by Kim *et al.* (2010); (5) the approach is extended to the more practical scenario of multiple spacecraft in different coordinate frames rather than the single coordinate framework of Kim and Mesbahi (2004), Kim *et al.* (2010) and Okoloko and Kim (2010). Due to the aforementioned contributions, this work provides a more practically oriented solution for constrained attitude control of multiple spacecraft systems than the previous works of Kim and Mesbahi (2004), Kim *et al.* (2010).

The rest of the chapter is organized as follows: In Section 4.2, the problem formulation for this work is presented. In Section 4.3, brief mathematical preliminaries are presented, and the solution technique and convergence analysis are provided in Section 4.4. Numerical simulations are given in Section 4.5, and concluding remarks follow in Section 4.6.

4.2 Problem Statement

Given a set of spacecraft, with initial states $x^i(t_0) \in \mathbb{R}^3$, $i = 1, \cdots, n$, initial attitude quaternions $q^i(t_0)$, and the Laplacian matrix of their communication graph $\mathbf{L}$, our concern is to drive $q^i(t)$ to a consensus attitude quaternion $q^c = q(t_f)$, while satisfying collision avoidance and norm constraints. Note that $q(t_f)$ needs not be known a priori to any of the spacecraft.

The problem stated above has two major parts: *consensus* and *collision avoidance*. The consensus part is basically that of driving the attitudes to a consensus attitude. The final consensus attitude is usually the centroid of the initial attitudes $q_{ave} = \frac{\frac{1}{n}\sum_{i=1}^{n} q^i(0)}{\|\frac{1}{n}\sum_{i=1}^{n} q^i(0)\|}$, meaning that each spacecraft should eventually point in the same direction. But by applying relative offset quaternion vectors, the consensus attitude can also be a desired formation attitude, e.g. each spacecraft can point at 15° away from each other about the z axis. By applying leader follower architectures, or leaderless architectures with set point control, the final consensus attitude can also be a set of desired final attitudes for each spacecraft, e.g. SC_1 can be made to point in a particular direction, while various offset angles from the attitude of SC_1 are defined for every other spacecraft. One may think that a standard consensus protocol such as (3.2.2) could be used directly to solve the consensus seeking part of the problem above. However, such a protocol violates the non-linearity of quaternion kinematics and the quaternion norm preserving constraint. Even though a protocol is found which accommodates the quaternion kinematics and the norm preserving constraints, its convergence analysis may not be as simple as (3.2.2).

For the original problem statement to be meaningful practically, the collision avoidance part cannot be excluded. Rotating spacecraft must avoid (angle-wise) collisions in order to reach their consensus attitude or form a desired formation. For example, a spacecraft which is rotating a photosensitive instrument from one attitude to another may need to avoid blinding celestial objects. As discussed in the previous works of Kim and Mesbahi (2004), Kim *et al.* (2010) and Okoloko and Kim (2010), this collision avoidance part is a computationally difficult Q-CAC problem. As the complexity of the Q-CAC problem is greatly affected by the number of LMI constraints, it may not be a good idea that the two parts of the problem statement are posed as a single optimization problem.

For the above reason, the following strategy will be proposed later in Section 4.4. Each part is posed as a separate optimization problem and solved simultaneously at each time step. While the consensus part computes a guidance command or consensus attitude trajectory for each spacecraft, the collision avoidance part decides whether it is safe to track the computed consensus trajectory, and generates a proper control input for each spacecraft to track the trajectory. If the consensus trajectory is not safe, the collision avoidance part computes a new set of quaternion vectors that avoid collision and asks the consensus part to compute a new consensus trajectory in the next step. This cycle repeats until consensus is achieved. Unlike the previous approaches in the literature, the guidance command computed by the consensus part satisfies the quaternion kinematics and the norm preserving constraints while guaranteeing the average consensus (see Section 4.4.1); and the control input generated by the collision avoidance part is decentralized and also valid in the multiple coordinate frame setting (see Section 4.4.2).

To illustrate the collision avoidance part (Q-CAC problem), we begin with a single spacecraft and a single obstacle vector. Denote spacecraft i by SC_i and $v^I_{cam}(t)$ by the unit camera vector in $\mathcal{F}^I_{SCi}$ corresponding to the SC_i's attitude $q^i(t)$. Let $v^I_{obs}(t)$ be the unit vector corresponding to the attitude quaternion of the obstacle to be avoided (see Figure 4.3). It is desired that the time evolution of the camera vector of SC_i from $v^I_{cam}(t_0)$ to $v^I_{cam}(t_f)$ should avoid $v^I_{obs}(t)$ at all times, while maintaining a minimum angular separation of ϕ. The requirement is thus that

$$\theta(t) \geq \phi \quad \forall t \in [t_0 \; t_f],$$

or

$$v^I_{cam}(t)^T v^I_{obs}(t) \leq \cos\phi \quad \forall t \in [t_0 \; t_f]. \tag{4.2.1}$$

Equation (4.2.1) is the same attitude constraint problem presented in (3.4.4).

A solution to this problem was provided by Kim and Mesbahi (2004), using the quaternion attitude constraints formulation developed by Ahmed *et al.* (1998). However, the solution was for a single spacecraft and single obstacle case, where v^I_{obs} and $v^I_{cam}(t)$ evolve from the same coordinate frame, and v^I_{obs} is static. Here we are interested in extending the previous avoidance solution to multiple spacecraft and solving it for spacecraft and dynamic $v^I_{obs}(t)$, where $v^I_{obs}(t)$ and $v^I_{cam}(t)$ are in different coordinate frames for each spacecraft. In the next section, we discuss the basic mathematical preliminaries to the solution provided in Section 4.4.

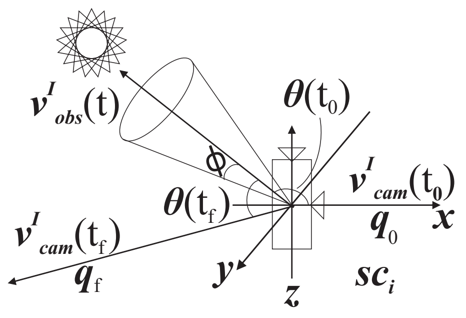


Figure 4.3: Constrained attitude control problem.

4.3 Mathematical Background

4.3.1 Rotational Dynamics Based on Quaternions

The attitude of a rigid body rotating in three dimensional space (such as a spacecraft or satellite) can be conveniently represented by unit quaternions (Hughes, 2004; Kuipers, 1999), which helps to forestall the problems of singularities inherent in using Euler angles. The quaternion is a four-element vector

$$q = [q_1\ q_2\ q_3 \mid q_4]^T.$$

Given that θ is the angle of rotation, the scalar part of the quaternion is $q_4 = \cos\frac{\theta}{2}$, and the axis of rotation is represented with a unit norm three element coordinate vector **a**. The vector part of the quaternion is $[q_1\ q_2\ q_3]^T \equiv \bar{q} = \mathbf{a}\sin\frac{\theta}{2}$ (Hughes, 2004). The difference between two quaternions q^1 and q^2 is given by

$$q^d = q^1 \ominus q^{-2} = q^1 \ominus \begin{bmatrix} -q_1^2 & -q_2^2 & -q_3^2 & q_4^2 \end{bmatrix}^T = Q^2 q^1,$$

where q^{-2} (or q^{2*}) is the conjugate of q^2, $\ominus$ is a quaternion difference operator, and

$$Q^i = \begin{bmatrix} q_4^i & q_3^i & -q_2^i & -q_1^i \\ -q_3^i & q_4^i & q_1^i & -q_2^i \\ q_2^i & -q_1^i & q_4^i & -q_3^i \\ q_1^i & q_2^i & q_3^i & q_4^i \end{bmatrix}.$$

Note that at the scalar level, the superscript notation used above does not indicate exponentiation but is an index instead. The rotational dynamics for the i^{th} quaternion are given as

$$\dot{q}^i = \frac{1}{2}\Omega^i q^i = \frac{1}{2}\Pi^i w^i,\ i = 1, \dots, n\,,$$

where

$$\Omega^i = \begin{bmatrix} 0 & w_3^i & -w_2^i & w_1^i \\ -w_3^i & 0 & w_1^i & w_2^i \\ w_2^i & -w_1^i & 0 & w_3^i \\ -w_1^i & -w_2^i & -w_3^i & 0 \end{bmatrix},\ \Pi^i = \begin{bmatrix} q_4^i & -q_3^i & q_2^i \\ q_3^i & q_4^i & -q_1^i \\ -q_2^i & q_1^i & q_4^i \\ -q_1^i & -q_2^i & -q_3^i \end{bmatrix}.$$

Using Euler's first-order discretization method, one has

$$q^i(k+1) = (\mathbf{I}_4 + \frac{\Delta t}{2}\Omega^i(k))q^i(k) = q^i(k) + \frac{\Delta t}{2}\Pi^i(k)w^i(k), \tag{4.3.1}$$

where $\mathbf{I}_n$ is the n-by-n identity matrix, and k and Δt are the discrete time index and the fixed step size. The rotational velocities w^i evolve according to

$$\begin{aligned} \begin{bmatrix} \dot{w}_1^i \\ \dot{w}_2^i \\ \dot{w}_3^i \end{bmatrix} &= \begin{bmatrix} ((J_2^i - J_3^i)w_2^i w_3^i + \tau_1^i)/J_1^i \\ ((J_3^i - J_1^i)w_3^i w_1^i + \tau_2^i)/J_2^i \\ ((J_1^i - J_2^i)w_1^i w_2^i + \tau_3^i)/J_3^i \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} 0 & \frac{J_2^i}{J_1^i}w_3^i & -\frac{J_3^i}{J_1^i}w_2^i \\ \frac{J_3^i}{J_2^i}w_3^i & 0 & -\frac{J_1^i}{J_2^i}w_1^i \\ \frac{J_1^i}{J_3^i}w_2^i & -\frac{J_2^i}{J_3^i}w_1^i & 0 \end{bmatrix}}_{\Upsilon^i} \begin{bmatrix} w_1^i \\ w_2^i \\ w_3^i \end{bmatrix} + \underbrace{\begin{bmatrix} 1/J_1^i & 0 & 0 \\ 0 & 1/J_2^i & 0 \\ 0 & 0 & 1/J_3^i \end{bmatrix}}_{(J^i)^{-1}} \begin{bmatrix} \tau_1^i \\ \tau_2^i \\ \tau_3^i \end{bmatrix}. \end{aligned} \tag{4.3.2}$$

Again, after the discretization

$$w^i(k+1) = \left(\mathbf{I}_3 + \Delta t \Upsilon^i(k)\right) w^i(k) + \Delta t (J^i)^{-1}\tau^i(k), \tag{4.3.3}$$

where J_j^i, w_j^i and τ_j^i, $j = 1, 2, 3$, are the moments of inertia, rotational velocities, and control torques, along the three principal axes j, for the i^{th} rigid body. Combining $(4.3.1) - (4.3.3)$ in stacked vector form, the discrete time dynamics evolve according to

$$\underbrace{\left[\begin{array}{c|c|c} -\Delta t (J^i)^{-1} & \mathbf{I}_3 & \mathbf{0}_{3\times 4} \\ \hline \mathbf{0}_{4\times 3} & -\Delta t \Pi^i(k+1)/2 & \mathbf{I}_4 \end{array}\right]}_{\mathbf{F}^i(k)} \underbrace{\begin{bmatrix} \tau^i(k) \\ w^i(k+1) \\ q^i(k+2) \end{bmatrix}}_{\mathbf{z}^i(k)} = \underbrace{\begin{bmatrix} (\mathbf{I}_3 + \Delta t \Upsilon^i(k))w^i(k) \\ q^i(k+1) \end{bmatrix}}_{\mathbf{y}^i(k)} \tag{4.3.4}$$

Determining τ that stabilizes the system is the typical task of controller synthesis for attitude stabilization. In order to include the dynamic system (4.3.4) in the semidefinite program, it is represented as a dynamic *constraint* parameterized as (Kim *et al.*, 2010)

$$\mathbf{F}^i(k)\mathbf{z}^i(k) = \mathbf{y}^i(k), \tag{4.3.5}$$

where $\mathbf{y}^i(k)$ is expanded as

$$\mathbf{y}^i(k) = \begin{bmatrix} J_1^i w_1^i(k) + \Delta t(J_2^i - J_3^i)w_2^i(k)w_3^i(k) \\ J_2^i w_2^i(k) + \Delta t(J_3^i - J_1^i)w_3^i(k)w_1^i(k) \\ J_3^i w_3^i(k) + \Delta t(J_1^i - J_2^i)w_1^i(k)w_2^i(k) \\ q^i(k+1) \end{bmatrix}.$$

4.3.2 Consensus Algorithm for Quaternions

Recall the concepts of consensus theory from Section 3.2. In order to extend protocol $(3.2.3)$ to attitude quaternions, the following protocol was proposed by Okoloko and Kim (2010):

$$\dot{\mathbf{q}}(t) = -\mathbf{P}(t)(\mathbf{q}(t) \ominus \mathbf{q}^{-off}), \tag{4.3.6}$$

where **P** is a Laplacian-like matrix and $\mathbf{q}(t) = [q^1(t)^T \; q^2(t)^T \; \cdots \; q^n(t)^T]^T$. More detailed discussions on $(4.3.6)$ follow in Section 4.4.

The synthesis of the $\mathbf{P}(t)$ matrix as a collective Laplacian-like matrix suggests a centralized implementation, however it is only developed here for the purpose of simplicity and analysis. Therefore for clarity, as we progress, whenever an equation or item is presented for the collection of vehicles, the decentralized individual equivalent for a single vehicle will also be presented.

4.4 Solution

In this section, a solution is developed for the problem stated in Section 4.2. The solution involves four intermediate steps: (i) synthesis of a consensus attitude for multiple spacecraft; (ii) formulation of Q-CAC in different coordinate frames; (iii) determination of obstacle vectors in different coordinate frames; and (iv) consensus based Q-CAC.

4.4.1 Synthesis of Consensus Attitude For Multiple Spacecraft

The problem under consideration can be coded as a set of LMI, and solved for optimal quaternions trajectories, using available optimization software tools, such as Sedumi (Sturm, 1998) and Yalmip (Lofberg, 2004). To this end, the objective function $(4.3.6)$ is augmented with an arbitrary number of constraints which are defined as we proceed.

Given the set of initial attitude quaternions in a column vector $\mathbf{q}(t_0)$, and the associated graph Laplacian $\mathbf{L}$, if one wants to use $\mathbf{L}$ directly to drive q^i to consensus, one can set $\Gamma = \mathbf{L} \otimes \mathbf{I}_4$ and then, replace $\mathbf{P}$ with Γ in $(4.3.6)$, so that $(4.3.6)$ drives q^i to a *consensus* value, in the sense that all of q^i actually converge to a single value at steady state. However, this final value obtained is not a unit quaternion, because the process is not quaternion norm preserving. If this consensus quaternion is normalized programmatically, or when the resulting quaternion trajectories are normalized at every time step, the resulting consensus quaternion deviates significantly from the true average quaternion, and therefore is erroneous.

To solve the above problem, the idea is to obtain $\mathbf{P}$ in $(4.3.6)$ which can help to preserve quaternion norm, or help to obtain a consensus quaternion that, when normalized, is an accurate average consensus quaternion. To obtain this $\mathbf{P}$, we begin by defining n positive semidefinite matrix variables, $\Lambda^1, \Lambda^2, \cdots, \Lambda^n$, for n quaternions, $\Lambda^i \in \mathcal{S}^4$. Here $\mathcal{S}^n$ denotes the set of $n \times n$ positive semidefinite matrices. Then, Λ^i and elements l_{ij} of $\mathbf{L}$ are used to build $\mathbf{P}$, which retains the Laplacian characteristics of $\mathbf{L}$, meaning that the rows of $\mathbf{P}$ sum to zero. To this end, $\mathbf{P}(t)$ is defined as

$$\mathbf{P}(t) = \underbrace{\begin{bmatrix} \Lambda^1(t) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \Lambda^n(t) \end{bmatrix}}_{\Lambda(t)} \underbrace{\begin{bmatrix} l_{11}\mathbf{I}_4 & \dots & l_{1n}\mathbf{I}_4 \\ \vdots & \ddots & \vdots \\ l_{n1}\mathbf{I}_4 & \dots & l_{nn}\mathbf{I}_4 \end{bmatrix}}_{\Gamma=\mathbf{L}\otimes\mathbf{I}_4},$$

where n is the number of vehicles.

The collective dynamics $(4.3.6)$ is useful for analysis and whenever a centralized implementation is desired. However, in this work we are interested in a decentralized implementation of the algorithm with intervehicle communication. The decentralized $\mathbf{P}^i$ for an individual vehicle i is built by block matrix multiplication of $\Lambda^i(t)$ and elements of the communication graph observed by i at time t. For example, for vehicle 1 one has

$$\begin{aligned}\mathbf{P}^1(t) &= \left[\Lambda^1(t)l_{11}\mathbf{I}_4 \;\; \Lambda^1(t)l_{12}\mathbf{I}_4 \cdots \Lambda^1(t)l_{1n}\mathbf{I}_4\right],\\ &= \left[n\Lambda^1(t) \;\; -\Lambda^1(t) \cdots -\Lambda^1(t)\right].\end{aligned}$$

Therefore for any i one has

$$\mathbf{P}^i(t) = \left[y\Lambda^i_1(t) \;\; -\Lambda^i_2(t) \cdots -\Lambda^i_y(t)\right],$$

where y is the number of vehicles in the communication neighborhood of vehicle i.

Neglecting offset quaternions, the continuous time individual dynamics for i is

$$\dot{q}^i(t) = -\mathbf{P}^i(t)\left[q_1^T(t)\; q_2^T(t) \;\cdots\; q_y^T(t)\right]^T,$$

where $q_1^T(t), q_2^T(t), \cdots, q_y^T(t)$ are the quaternions of the vehicles that i is communicating with at time t.

Next, some theoretical properties of $(4.3.6)$ are given, assuming full connectivity of the graph of $\mathbf{L}$. For the purpose of analysis, it is required to prove that $\mathbf{q}$ in $(4.3.6)$ is indeed driven to consensus. First, take a closer look at the structure of $\mathbf{P}$: note that $\Lambda^i(t) > 0\ \forall i$. It therefore follows that $\Lambda(t) > 0$ and, since $\mathbf{L}$ is symmetric, it follows that Γ is also symmetric. The rows of $\mathbf{L}$ sum to 0, therefore $\mathbf{L}$ has positive eigenvalues except for one zero eigenvalue, which also implies that Γ has positive eigenvalues except for four zero eigenvalues.

Theorem 4.1 *The time varying system* $(4.3.6)$ *achieves consensus assuming connectivity of* $\mathbf{L}$.

Proof: Without loss of generality, assume that $\mathbf{q}^{off} = \mathbf{0}$[1]. First note that if $\mathbf{q}$ belongs to the *consensus space* $\mathcal{C} = \{\mathbf{q}\,|\,q^1 = q^2 = \cdots = q^n\}$, then $\dot{\mathbf{q}} = \mathbf{0}$. In fact, $\mathcal{C}$ is the nullspace of $\mathbf{P}(t)$, i.e. the set of all $\mathbf{q}$ such that $\mathbf{P}(t)\mathbf{q} = \mathbf{0}$. Thus, once $\mathbf{q}$ enters $\mathcal{C}$ it stays there. Assume that $\mathbf{q} \notin \mathcal{C}$, and consider a Lyapunov candidate function $V = \mathbf{q}^T\Gamma\mathbf{q}$. Note that $V > 0$

[1]This is equivalent to $(q^{off})^i = [0\;0\;0\;1]^T\ \forall\, i$.

unless $\mathbf{q} \in \mathcal{C}$. Then,

$$\begin{aligned}\dot{V} =& \mathbf{q}^T \Gamma \dot{\mathbf{q}} + \dot{\mathbf{q}}^T \Gamma \mathbf{q},\\ =& -\mathbf{q}^T \Gamma \mathbf{P}(t)\mathbf{q} - \mathbf{q}^T \mathbf{P}(t)^T \Gamma \mathbf{q},\\ =& -2\mathbf{q}^T \Gamma \Lambda(t) \Gamma \mathbf{q} = -2\mathbf{y}^T \Lambda(t) \mathbf{y} < 0,\end{aligned}$$

where $\mathbf{y} = \Gamma \mathbf{q} \neq \mathbf{0}$ for $\mathbf{q} \notin \mathcal{C}$. Therefore, $\mathbf{q}$ approaches a point in $\mathcal{C}$ as t tends to ∞. This proves the claim.

Next, to define dynamics constraints, one begins by applying Euler's first-order discretization to (4.3.6):

$$\dot{\mathbf{q}}_k \approx \frac{\mathbf{q}_{k+1} - \mathbf{q}_k}{\Delta t}.$$

Assuming zero offset, one has

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \Delta t \dot{\mathbf{q}}_k = \mathbf{q}_k - \Delta t \mathbf{P}(t)\mathbf{q}_k. \tag{4.4.1}$$

The discrete time dynamics for vehicle i is therefore given as

$$q^i_{k+1} = q^i_k - \Delta t \mathbf{P}^i(t) \left[q_1^T(k) \; q_2^T(k) \; \cdots \; q_y^T(k) \right]^T.$$

The dynamics constraint (4.4.1) is sufficient to obtain a consensus attitude that, when normalized, is the accurate average consensus value. However, in order to include avoidance constraints directly without having to normalize $q^i(t)$ every time step after solving (4.4.1), an alternative is to include a constraint that enforces $\|q^i(t)\| = 1$. To satisfy this constraint for each i and $t \in [t_0, t_f]$, it is sufficient to enforce the following linear constraint

$$\mathbf{q}_k^T(\mathbf{q}_{k+1} - \mathbf{q}_k) = 0, \tag{4.4.2}$$

or

$$(q^i)_k^T (q^i_{k+1} - q^i_k) = 0,$$

for a decentralized implementation.

Equation (4.4.2) is essentially the discrete time version of $\mathbf{q}(t)^T \dot{\mathbf{q}}(t) = 0$ or $q^i(t)^T \dot{q}^i(t) = 0$ which guarantees $\mathbf{q}(t)^T \mathbf{q}(t) = n$ or $q^i(t)^T q^i(t) = 1$ as long as $\|q^i(0)\| = 1$ for each i.

Next, we discuss the method that Kim *et al.* (2010) used to enforce attitude constraints, which was also adopted by Okoloko and Kim (2010). The method is presented here again for clarity, to show how the attitude constraints were represented as LMI. In the work of Okoloko and Kim (2010), to enforce attitude constraints, we represent (4.2.1) as a LMI, following the procedure of Kim *et al.* (2010).

Let $\mathbf{v}(t) = \begin{bmatrix} v^I_{cam}(t) \\ v^I_{obs}(t) \end{bmatrix}$. Also let $\mathfrak{F}$ be a function or mapping that transforms a unit vector to the equivalent unit quaternion, let $v^1 = v^I_{cam}(t)$ and $v^2 = v^I_{obs}(t)$. Then

$$\begin{aligned} q^1 =& \mathfrak{F}_1(v^1), \\ q^2 =& \mathfrak{F}_2(v^2). \end{aligned}$$

For some inverse transformations $\mathfrak{H}_1$ and $\mathfrak{H}_2$ on $\mathfrak{F}$,

$$\begin{aligned} v^1 =& \mathfrak{H}_1(q^1), \\ v^2 =& \mathfrak{H}_2(q^2). \end{aligned}$$

This therefore implies that

$$(v^1)^T v^2 = (\mathfrak{H}_1(q^1))^T(\mathfrak{H}_2(q^2)) = (q^1)^T \mathfrak{H}_1^T \mathfrak{H}_2 q^2 \leq \cos\phi. \tag{4.4.3}$$

Let $q = [(q^1)^T \ (q^2)^T]^T$, then (4.4.3) can be written as

$$q^T \underbrace{\begin{bmatrix} 0 & \frac{1}{2}\mathfrak{H}_1^T\mathfrak{H}_2 \\ \frac{1}{2}\mathfrak{H}_2^T\mathfrak{H}_1 & 0 \end{bmatrix}}_{\mathbf{H}} q \leq \cos\phi.$$

For some positive real number μ larger than the largest absolute value of the eigenvalues of $\mathbf{H}$

$$q^T \left(\mu \mathbf{I}_8 + \begin{bmatrix} 0 & \mathfrak{H}_1^T\mathfrak{H}_2 \\ \mathfrak{H}_2^T\mathfrak{H}_1 & 0 \end{bmatrix} \right) q \leq 2(\mu + \cos\phi). \tag{4.4.4}$$

Let $\mathbf{W} = \left(\mu \mathbf{I}_8 + \begin{bmatrix} 0 & \mathfrak{H}_1^T\mathfrak{H}_2 \\ \mathfrak{H}_2^T\mathfrak{H}_1 & 0 \end{bmatrix} \right)^{-1}$, then from the Schur complement formula, the LMI equivalent of (4.4.4) is

$$\begin{bmatrix} 2(\mu + \cos\phi) & q^T \\ q & \mathbf{W} \end{bmatrix} \geq 0. \tag{4.4.5}$$

The relationship between $q^i(t)$ and $v^i(t)$ for any i, can be expressed as

$$v^i(t) = \begin{bmatrix} 2(q_1^i)^2(t) + 2(q_4^i)^2(t) - 1 \\ 2q_1^i(t)q_2^i(t) + 2q_3^i(t)q_4^i(t) \\ 2q_1^i(t)q_3^i(t) - 2q_2^i(t)q_4^i(t) \end{bmatrix}. \tag{4.4.6}$$

It remains to determine the matrix $\mathbf{W}_l^i$ for each $v_l^i(t)$ $(l = 1, 2, 3)$ in (4.4.6). Expressing the rows of (4.4.6) as quadratic forms, the elements of $v^i(t)$ are

$$v_1^i(t) + 1 = q^i(t)^T \mathbf{W}_1^i q^i(t), \tag{4.4.7}$$

$$v_2^i(t) = q^i(t)^T \mathbf{W}_2^i q^i(t), \tag{4.4.8}$$

$$v_3^i(t) = q^i(t)^T \mathbf{W}_3^i q^i(t), \tag{4.4.9}$$

where

$$\mathbf{W}_1^i = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}, \mathbf{W}_2^i = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \mathbf{W}_3^i = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}.$$

Expressing (4.4.7) as LMI, one has

$$\mathbf{L}_{1+}^i(t) := \begin{bmatrix} \mu_{1_{\mathbf{W}_1^i}} - v_1^i(t) - 1 & q^i(t)^T \\ q^i(t) & (\mu_{1_{\mathbf{W}_1^i}} I_4 - \mathbf{W}_1^i)^{-1} \end{bmatrix} \geq 0, \tag{4.4.10}$$

$$\mathbf{L}_{1-}^i(t) := \begin{bmatrix} \mu_{2_{\mathbf{W}_1^i}} + v_1^i(t) + 1 & q^i(t)^T \\ q^i(t) & (\mu_{2_{\mathbf{W}_1^i}} I_4 + \mathbf{W}_1^i)^{-1} \end{bmatrix} \geq 0, \tag{4.4.11}$$

where $\mu_{1_{\mathbf{W}_l^i}}$ and $\mu_{2_{\mathbf{W}_l^i}}$ are chosen to be larger than the largest eigenvalues of $\mathbf{W}_l^i$, and $-\mathbf{W}_l^i$ $(l = 1, 2, 3,\ i = 1, \cdots, n)$, respectively.

Bringing it all together, the optimization problem of finding a feasible attitude trajectory may be posed as an SDP, as follows. Given the set of initial quaternions $q^i(t_0)$, $(i = 1, \cdots, n)$, and the plant equation (4.3.6), find a feasible sequence of quaternions that converges to a consensus quaternion $q^c = q(t_f)$ at steady state, and satisfies the following constraints:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \Delta t\dot{\mathbf{q}}_k = \mathbf{q}_k - \Delta t\mathbf{P}(t)\mathbf{q}_k,$$

$$\mathbf{q}_k^T(\mathbf{q}_{k+1} - \mathbf{q}_k) = 0,$$

$$\begin{bmatrix} 2(\mu + \cos\phi) & q \\ q^T & \mathbf{W} \end{bmatrix} \geq 0,$$

$$\mathbf{L}_{1+}^i(t) := \begin{bmatrix} \mu_{1_{\mathbf{W}_1^i}} - v_1^i(t) - 1 & q^i(t)^T \\ q^i(t) & (\mu_{1_{\mathbf{W}_1^i}} I_4 - \mathbf{W}_1^i)^{-1} \end{bmatrix} \geq 0,$$

$$\mathbf{L}_{1-}^i(t) := \begin{bmatrix} \mu_{2_{\mathbf{W}_1^i}} + v_1^i(t) + 1 & q^i(t)^T \\ q^i(t) & (\mu_{2_{\mathbf{W}_1^i}} I_4 + \mathbf{W}_1^i)^{-1} \end{bmatrix} \geq 0,$$

This strategy was indeed attempted by Okoloko and Kim (2010). However, due to computational complexity associated with solving the LMI associated with (4.4.7), it was difficult to implement attitude avoidance constraints directly in the consensus framework. In fact, when more than two avoidance constraints are defined, the algorithm tends to break down.

Moreover, in reality, multiple spacecraft navigate in their own different coordinate frames, not like the single coordinate framework presented by Kim and Mesbahi (2004), Kim *et al.* (2010), and Okoloko and Kim (2010). However, excluding (4.4.6) and the LMI associated with (4.4.7)-(4.4.9) from the framework, a solution can be developed by using the same framework to solve a different problem, that of constrained coordinated navigation on a sphere.

The conclusion is, clearly, that the avoidance strategy developed by Kim *et al.* (2010), and adopted by Okoloko and Kim (2010) is not practicable for constrained attitude control of multiple spacecraft using quaternions. This motivates a new formulation, which is the strategy developed in Sections 4.4.2-4.4.4 (Okoloko and Kim, 2012*b*,*a*).

4.4.2 Q-CAC in Different Coordinate Frames

Let us revisit (4.2.1). At any time $t \in [t_0, t_f]$, a rigid body attached to the body of the i^{th} spacecraft, e.g. a camera, whose pointing direction is $v_{cam_i}^I$, measured in inertial frame, can be transformed to the spacecraft body frame (and vice versa) by

$$v^B_{cam_i}(t) = \mathbf{R}_i(t) v^I_{cam_i}(t), \tag{4.4.12}$$

where

$$\mathbf{R}_i(t) = ((2q^i_4(t))^2 - 1)\mathbf{I}_3 + 2\bar{q}^i(t)\bar{q}^i(t)^T - 2q^i_4(t)\bar{q}^i(t)^\times,$$

is the rotation matrix corresponding to the attitude quaternion $q^i(t)$ of SC_i at time t (Hughes, 2004; Ren and Beard, 2004) , and $\bar{q}^i(t)^\times$ is the antisymmetric matrix, defined as

$$\bar{q}^i(t)^\times = \begin{bmatrix} 0 & -q^i_3(t) & q^i_2(t) \\ q^i_3(t) & 0 & -q^i_1(t) \\ -q^i_2(t) & q^i_1(t) & 0 \end{bmatrix}.$$

Suppose there is only one spacecraft, SC_i, and one camera, $v^I_{cam_i}$, and m (possibly time-varying) obstacles, $v^I_{obs_i \cdot j}$ $(j = 1, \ldots, m)$, to be avoided. Here, all $v^I_{obs_i \cdot j}$ are defined in $\mathcal{F}^I_{SCi}$. Then, following Ahmed *et al.* (1998), the attitude constraints described by (4.2.1) can be written as

$$q^i(t)^T \tilde{A}^i_j(t) q^i(t) \leq 0. \tag{4.4.13}$$

Its LMI equivalent (Kim and Mesbahi, 2004) is

$$\begin{bmatrix} \mu & q^i(t)^T \\ q^i(t) & (\mu\mathbf{I}_4 + \tilde{A}^i_j(t))^{-1} \end{bmatrix} \geq 0, \tag{4.4.14}$$

where μ is chosen such that $\mu\mathbf{I}_4 + \tilde{A}^i_j(t)$ is positive definite, and

$$\tilde{A}^i_j(t) = \begin{bmatrix} A_j(t) & b_j(t) \\ b_j(t)^T & d_j(t) \end{bmatrix} \in \mathbb{R}^{4\times 4},$$

$$A_j(t) = v^B_{cam_i}(t)(v^I_{obs_i \cdot j}(t))^T + v^I_{obs_i \cdot j}(t)(v^B_{cam_i}(t))^T - ((v^B_{cam_i}(t))^T v^I_{obs_i \cdot j}(t) + \cos\theta)\mathbf{I}_3,$$

$$b_j(t) = -v^B_{cam_i}(t) \times v^I_{obs_i \cdot j}(t),$$

$$d_j(t) = (v^B_{cam_i}(t))^T v^I_{obs_i \cdot j}(t),$$

for $j = 1, \cdots, m$.

Eq. (4.4.13) defines the set of attitude quaternions $q^i(t)$ such that $v^I_{cam_i}(t)^T v^I_{obs_i \cdot j}(t) \geq \phi$ for all $t \in [t_0, t_f]$. Therefore at each time step, (4.4.13) can be used to find a collision-free $v^I_{cam_i}(t)$. In many practical circumstances, however, another obstacle vector $v^I_{obs_j}(t)$ defined in $\mathcal{F}^I_{SCj}$ also needs to be avoided by SC_i. For example, as already mentioned before, a thruster vector emanating from SC_j must be avoided by SC_i. In order to address such a practical issue, we develop a mechanism to calculate $v^I_{obs_i \cdot j}$ (defined in $\mathcal{F}^I_{SCi}$) corresponding to $v^I_{obs_j}$ (defined in $\mathcal{F}^I_{SCj}$)[2]. This mechanism is essentially to determine the intersection point of $v^I_{obs_j}(t)$ with the sphere of radius r, centered on SC_i. If indeed there is such an intersection, the intersection point defines $v^I_{obs_i \cdot j}$ which can be used to define an attitude constraint represented as (4.4.13) and so avoided by SC_i.

To illustrate, suppose SC_1 and SC_2 are in their different coordinate frames relative to Earth as shown in Figure 4.4. A thruster attached to the body frame of SC_1 is at $v^I_{obs_1}$. The circles around SC_1 and SC_2 are the spheres from which their attitudes evolve. If the spacecraft are close enough to each other, then vector $v^I_{obs_1}$ may intersect the sphere of SC_2, as is shown. The point of intersection defines $v^I_{obs_2 \cdot 1}$ in the frame of SC_2, and it is desired that, as SC_2 changes its attitude from q_0 to q_f, $v^I_{cam_2}$ must avoid the cone created around $v^I_{obs_2 \cdot 1}$ by at least ϕ, $\forall t \in [t_0, t_f]$.

Note that the above strategy nicely *decentralizes* the problem in question. In other words, once those intersection points are determined, each spacecraft can solve its own optimization problem independently of states of neighbouring spacecraft. The computational advantage gained from this strategy will be demonstrated in Section 4.5.7.

4.4.3 Determination of $v^I_{obs_i \cdot j}(t)$

Given two spacecraft, SC_i in $\mathcal{F}^I_{SCi}$ and SC_j in $\mathcal{F}^I_{SCj}$, one can easily determine the intersection between a vector emanating from $\mathcal{F}^I_{SCj}$ and the unit sphere centered on $\mathcal{F}^I_{SCi}$ by using results from computational geometry (Eberly, 2001; van den Bergen, 2004). A point $p = [p_x \; p_y \; p_z]^T$ on a line segment, originating at p_1 and terminating at p_2, can be tested

[2]For simplicity, we assume that there is only one obstacle vector originated from SC_j for SC_i to avoid. Thus in this context, $v^I_{obs_i \cdot j}$ means the obstacle vector originated from the rotating frame of SC_j but defined in $\mathcal{F}^I_{SCi}$.

for intersection with a sphere centered at p_3 with radius r (in this case, r is set to 1), as described by Eberly (2001). Thus for any $v^I_{obs_j}(t)$ in $\mathcal{F}^I_{SCj}$, if an intersection point $p(t)$ exists at time t with the sphere centred on $\mathcal{F}^I_{SCi}$ with radius r then $v^I_{obs_i \cdot j}(t) = p(t)$; otherwise, one can set $v^I_{obs_i \cdot j}(t) = -v^I_{cam_i}(t)$, so that no constraints violation occurs. The value of r must depend on the size of the rigid body.

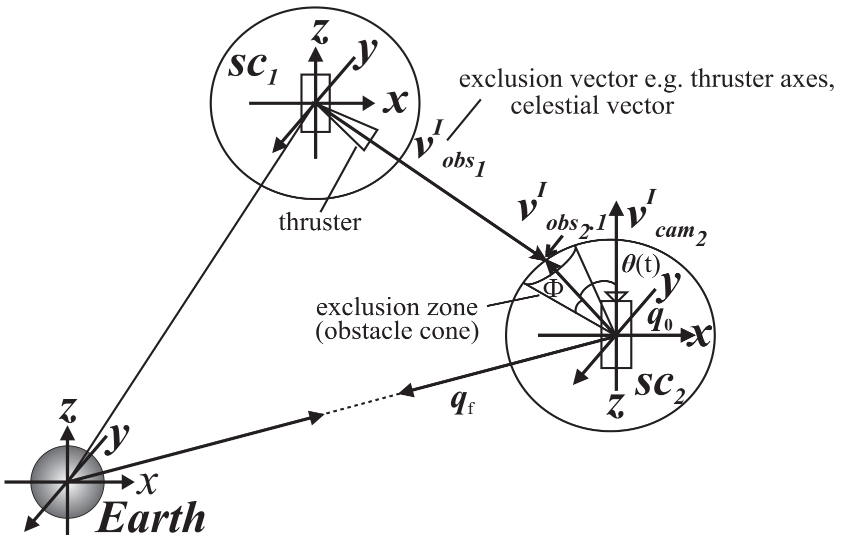

sc1
z
y
x
exclusion vector e.g. thruster axes, celestial vector
v^I obs1
thruster
v^I obs2.1
v^I cam2
θ(t)
Φ
exclusion zone (obstacle cone)
y
q0
x
sc2
z
qf
z
y
x
Earth

Figure 4.4: Q-CAC problem in different frames.

4.4.4 Consensus Based Q-CAC

Taking no consensus into consideration, the problem stated in Section 4.3 can be cast as an optimization problem augmented with a set of LMI constraints, and solved for optimal quaternions trajectories. In fact, we consider the algorithm in discrete time $k = 0, \cdots, N$ ($N\Delta t = t_f$). Given the initial and desired final attitude and angular velocity of SC_i, $(w^i(0), q^i(0)), (w^i(N), q^i(N))$, the optimization problem may be posed as follows:

$$\min_{w^i \tau^i} \left\|w^i(k+1) - w^i(N)\right\|^2 + \left\|q^i(k+2) \ominus q^{-i}(N)\right\|^2$$

subject to

$$\mathbf{F}^i(k)\mathbf{z}^i(k) = \mathbf{y}^i(k),$$
$$\left|w^i(k)\right| < w_{max}, \left|\tau^i(k)\right| < \tau_{max},$$
$$\mathbf{z}^i(k)^T H^i_j(k)\mathbf{z}^i(k) \leq 0,$$

where $j = 1, \cdots, m$ and

$$H^i_j(k) = \begin{bmatrix} \mathbf{0}_{6\times 6} & \mathbf{0}_{6\times 4} \\ \mathbf{0}_{4\times 6} & \tilde{A}^i_j(k) \end{bmatrix}$$

is defined so that $q^i(k+2)^T \tilde{A}^i_j(k) q^i(k+2) = \mathbf{z}^i(k)^T H^i_j(k)\mathbf{z}^i(k)$.

Let us now consider taking consensus into account. Unlike the previous Q-CAC problem, the consensus attitude $q^1(N) = \cdots = q^n(N)$ is unknown a priori in this context. There are two ways of going about this consideration. The first way is that one propagates $(4.3.6)$ and solves the Q-CAC problem simultaneously at every time step. In this case, for each time k one quickly calculates future quaternions, say up to step $k + \beta$, of each SC_i by synthesizing $\mathbf{P}(t)$ and propagating $(4.3.6)$. Among the future quaternions, *safe* quaternions q^i_{safe} $(i = 1, \cdots, n)$ which satisfy all the attitude or cone avoidance constraints are identified, then $q^i(N) = q^i_{safe}$ $(i = 1, \cdots, n)$ is set. Each spacecraft then solves the Q-CAC optimization problem stated above to generate a proper control torque τ^i. Once $q^i(N)$ $(i = 1, \cdots, n)$ is assumed, the same process is repeated. The second way is actually the same as the first one with sufficiently large β. In fact, one first computes a consensus attitude q^c for all SC_i using $(4.3.6)$, then solves the Q-CAC optimization problem with $q^i(N) = q^c$ [3]. While the second way will run faster and is simpler to implement, the first way is more suitable for real-time implementation as it accounts for time-varying communication topologies.

[3]Needless to say, q^c must be a safe quaternion.

4.5 Simulation Results

In this section, six simulation results are presented. The first three are for attitude synchronization in the same coordinate frame discussed in Section 4.4.1 (Okoloko and Kim, 2010), while the last three are for attitude synchronization in different coordinate frames discussed in Sections 4.4.2-4.4.4. A performance comparison of the proposed consensus based Q-CAC with $\beta = 2$ for different coordinate frames, with the approach for the same coordinate frame, and the former approach by Kim *et al.* (2010), is also presented.

The first experiment is to test attitude rendezvous, i.e. convergence without avoidance constraints, and the second experiment is for convergence with inter-vehicle collision avoidance constraints, while the third experiment is for attitude formation acquisition.

The fourth experiment is to test the dynamic avoidance in different coordinate frames for two spacecraft engaging in attitude manoeuvres, while consensus seeking is suppressed. In the fifth and sixth experiments, three spacecraft perform attitude manoeuvres to a consensus attitude while a sensitive instrument attached to each spacecraft avoids the plumes emanating from the thrusters of the two other spacecraft. Note that Sedumi (Sturm, 1998) and Yalmip (Lofberg, 2004) were used for solving all the optimization problems in this section. The simulations were done with Matlab R2009a on an Intel Core(TM)2 Duo P8600 @ 2.40GHz with 2 GB RAM, running Windows 7.

4.5.1 Attitude Rendezvous without Avoidance in the same Coordinate Frame

In this experiment four spacecraft converge to a common attitude, without avoidance constraints. The initial quaternions are $q_0^1 = [0.6882\ 0.0340\ 0.5727\ 0.4441]^T$, $q_0^2 = [0.7146\ -0.3404\ -0.3689\ -0.4872]^T$, $q_0^3 = [0.3149\ -0.4823\ -0.6478\ 0.4986]^T$, $q_0^4 = [-0.5384\ -0.3602\ 0.2889\ 0.7049]^T$. The resulting attitude trajectories for this test are shown in Figure 4.5. The graph on the right in the figure shows the convergence of the individual quaternion elements. Average consensus is demonstrated as the final value of convergence is $q^c = q(t_f) = [0.5837\ -0.5687\ -0.0768\ 0.5744]^T$, and this is the normalized average of the initial quaternion values. When the positions of the spacecraft are transformed to different coordinates on the surface of the sphere, all of them finally point in the same direction, as shown in Figure 4.6.

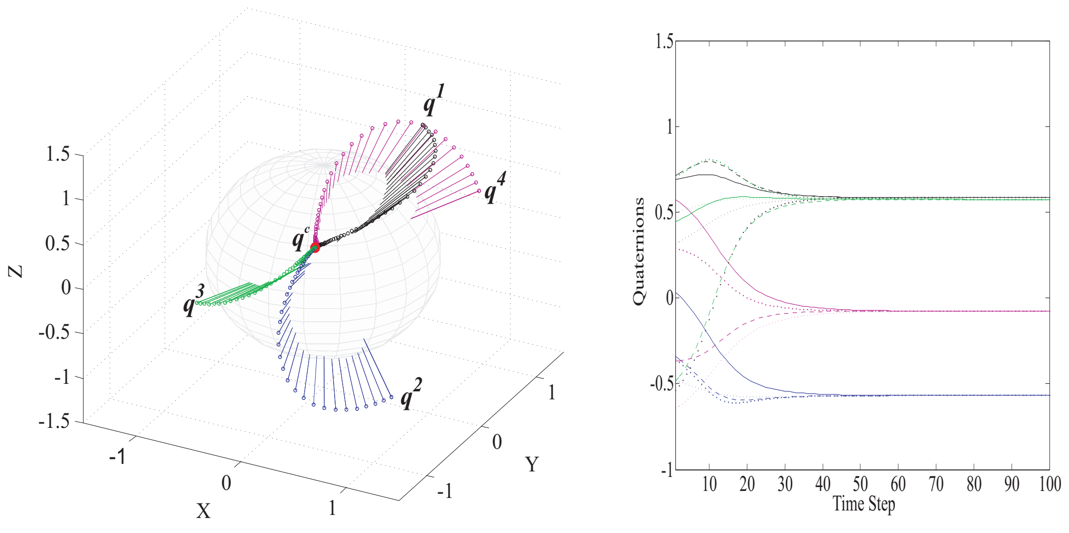


Figure 4.5: Four-spacecraft attitude rendezvous without avoidance constraints.

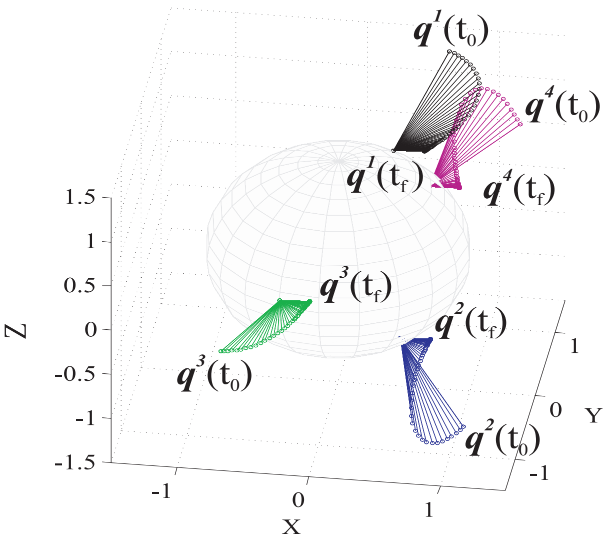


Figure 4.6: Four-spacecraft attitude rendezvous without avoidance constraints in different coordinate frames.

4.5.2 Convergence with Collision Avoidance in the same Coordinate Frame

In this experiment two spacecraft SC_1 and SC_4 are expected to avoid each other's attitude by a minimum angle of $60°$, while converging to consensus. This means that the final attitudes at steady state should be $60°$ apart. The initial quaternions are $q_0^1 = [0.6882\ 0.0340\ 0.5727\ 0.4441]^T$, $q_0^4 = [-0.5384\ -0.3602\ 0.2889\ 0.7049]^T$. The resulting attitude trajectories are shown in Figure 4.7 (left) and the avoidance graph is shown in Figure 4.7 (right).

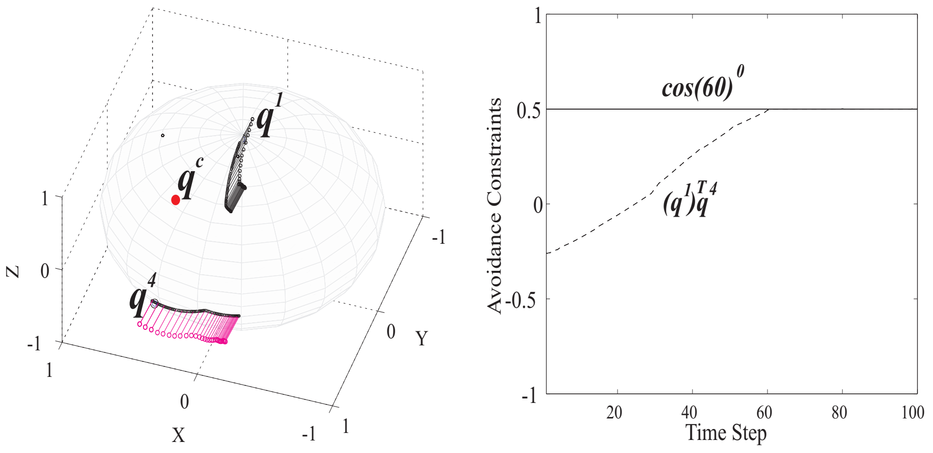


Figure 4.7: Two-spacecraft attitude reconfiguration with avoidance constraints in single coordinate frame (left), and the avoidance constraints graph (right).

4.5.3 Attitude Formation Acquisition in the same Coordinate Frame

In this experiment four spacecraft converge to attitudes that are relatively spaced by angular offsets $\phi^{off} = 30°$ about the z axis. The offset quaternions are $(q^1)^{off} = [1\ 0\ 0\ 0]^T$, $(q^2)^{off} = [0.9659\ 0\ 0\ 0.2588]^T$, $(q^3)^{off} = [0.8660\ 0\ 0\ 0.5]^T$, $(q^4)^{off} = [0.7071\ 0\ 0\ 0.7071]^T$. The initial quaternions are $q_0^1 = [0.6882\ 0.0340\ 0.5727\ 0.4441]^T$, $q_0^2 = [0.7146\ -0.3404\ -0.3689\ -0.4872]^T$, $q_0^3 = [0.3149\ -0.4823\ -0.6478\ 0.4986]^T$, $q_0^4 = [-0.5384\ -0.3602\ 0.2889\ 0.7049]^T$. At steady state the final quaternions were $q_f^1 = [0.2487\ -0.7761\ -0.3373\ 0.4712]^T$, $q_f^2 = [0.4411\ -0.6853\ -0.2038\ 0.5425]^T$, $q_f^3 = [0.6034\ -0.5478\ -0.0565\ 0.5767]^T$, $q_f^4 = [0.7246\ -0.3729\ 0.0947\ 0.5717]^T$. The attitude trajectories are in Figure 4.8, and the points of convergence of the equivalent unit vectors are shown in Figure 4.9.

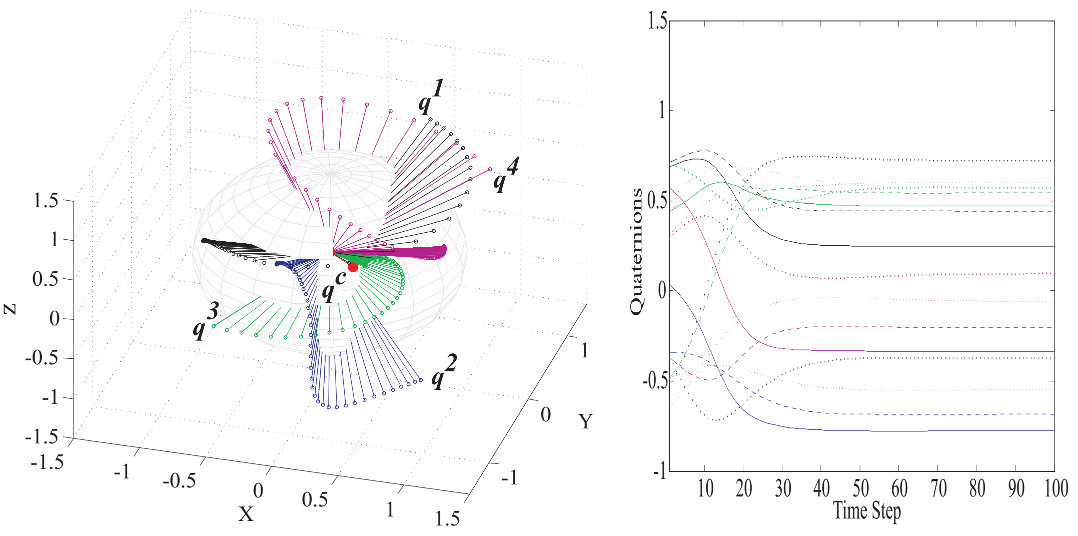


Figure 4.8: Four-spacecraft attitude formation acquisition.

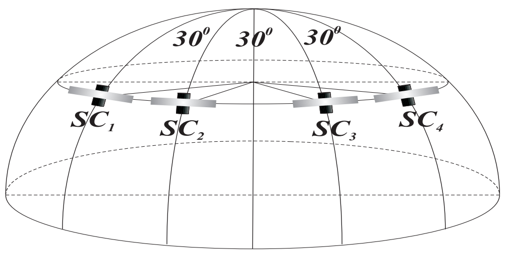


Figure 4.9: Convergence of the equivalent unit vectors for 30 degrees offset.

An interesting observation about this experiment is that, depending on the initial attitude quaternions, when a relative offset attitude vector is specified for attitude spacing along the z axis, the spacecraft converge to a common latitude along longitudes (great circles), separated by the offset angles, no matter their final attitudes. When a relative offset attitude is specified along the x or y axes, the spacecraft converge to a common longitude along latitudes. The cause of this unexpected behaviour has not been explored in this present research because it will require significant research effort to unravel and explain, and we hope to consider this as a part of future research. However, the strategy is useful for control along latitudes and longitudes, and can be utilized for coordinated stabilization of the collective motion of multiple aerospace vehicles navigating on the surface of a spheroid planet, such as Earth. A new method for constrained coordinated control of multiple vehicles on a sphere is proposed in Chapter 5.

4.5.4 Dynamic Avoidance in Different Coordinate Frames

In this experiment SC_1 and SC_2 are attempting a reconfiguration to Earth (either changing orientation to Earth or pointing an instrument to Earth). The initial quaternions of SC_1 and SC_2 are $q_0^1 = q_0^2 = [0\ 0\ 0\ 1]^T$. At steady state, the final quaternions of SC_1 and SC_2 are $q_f^1 = [0.2269\ 0.0421\ -0.9567\ 0.1776]^T$ and $q_f^2 = [0\ 0\ 0.9903\ 0.1387]^T$. Three thrusters of SC_1 in $\mathcal{F}_{SC1}^I$ are at $v_{obs_1 \cdot 1}^I = [-0.2132\ -0.0181\ -0.9768]^T$, $v_{obs_1 \cdot 2}^I = [0.314\ 0.283\ -0.906]^T$, $v_{obs_1 \cdot 3}^I = [-0.112\ -0.133\ -0.985]^T$. A single thruster of SC_2 in $\mathcal{F}_{SC2}^I$ is at $v_{obs_2}^I = [0.02981\ 0.0819\ 0.9962]^T$. It is desired that $v_{obs_2}^I$ should avoid $v_{obs_1 \cdot 1}^I$ by $50°$, and also avoid $v_{obs_1 \cdot 2}^I$ and $v_{obs_1 \cdot 3}^I$ by $30°$, while both spacecraft are maneuvering to their desired final attitudes. The results are shown by the trajectories in Figure 4.10 and avoidance graph in Figure 4.11. It can be observed that SC_2 cannot reconfigure to the desired $q^2(t_f)$ or q_f^2, due to the avoidance constraints. Note that in this experiment SC_1 and SC_2 positions have been chosen so that by the time SC_1 completes its reconfiguration, $v_{obs_1 \cdot 2}^I$ is resting in a position on SC_2 such that SC_2 cannot reconfigure exactly to the desired final quaternion without violating the avoidance constraints.

This experiment demonstrates that the avoidance constraint is superior to the desired final quaternion constraint, when both constraints are in conflict. The final quaternion is assumed by the minimization action, unlike the avoidance constraints which must be satisfied all the time. In this sense, the final quaternion constraint is a soft constraint, whereas the avoidance constraints are hard constraints. As can be seen, due to the satisfaction of the avoidance constraints, SC_2 cannot reconfigure exactly to the desired q_f^2, in which case it becomes necessary to change the position of either SC_2 or SC_1. The sudden jumps to and from -1 in Figure 4.11 indicate times when any of $v_{obs_1 \cdot 1}^I$, $v_{obs_1 \cdot 2}^I$, $v_{obs_1 \cdot 3}^I$ loses intersection with the sphere of SC_2, and therefore $v^I_{obs_2 \cdot k}$ $(k = 1, 2, 3)$ were replaced with $-v^I_{obs_2}$, as proposed in Section 4.4.3.

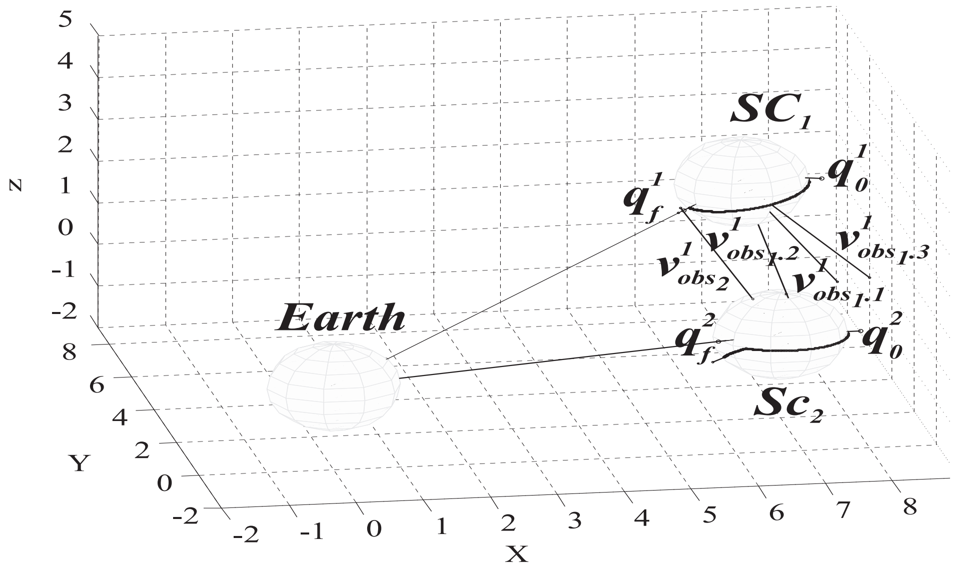


Figure 4.10: Dynamic avoidance in different coordinate frames.

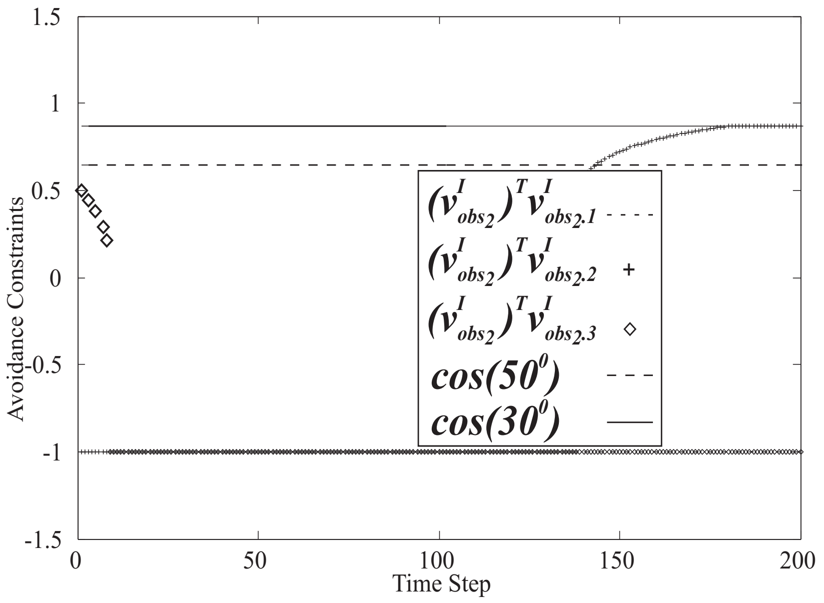


Figure 4.11: Avoidance constraints graph for Figure 4.10.

4.5.5 Consensus Based Dynamic Avoidance in Different Coordinate Frames

In this experiment SC_1, SC_2 and SC_3 will manoeuvre to a consensus attitude. A set of initial quaternions were randomly generated as $q_0^1 = [-0.5101\ 0.6112\ -0.3187\ -0.5145]^T$, $q_0^2 = [-0.9369\ 0.2704\ -0.1836\ -0.124]^T$ and $q_0^3 = [0.1448\ -0.1151\ 0.1203\ 0.9753]^T$. The origins of $\mathcal{F}^I_{SC1}$, $\mathcal{F}^I_{SC2}$ and $\mathcal{F}^I_{SC3}$ are $[-2\ 0\ 2]^T$, $[0.5\ 0\ 2]^T$ and $[3\ 0\ 2]^T$. A sensitive instrument (e.g. camera) $v^I_{cam_i}$ attached to each SC_i is pointing in the direction of its initial attitude quaternion. For simplicity, it is assumed that each spacecraft has only one thruster pointing to the opposite (rear) of its initial attitude. It is desired that the sensitive instrument avoids the thruster plumes emanating from each of the two other spacecraft by an angle of $30°$, during the period of the manueuvre. The spacecraft are positioned so that there is the possibility of intersection of the thrusters of SC_1 and SC_3, with SC_2, and also the thruster of SC_2 may negatively affect SC_1 or SC_3 at any time. The solution trajectories are shown in Figure 4.12, and the avoidance graph in Figure 4.13 shows that constraints are not violated.

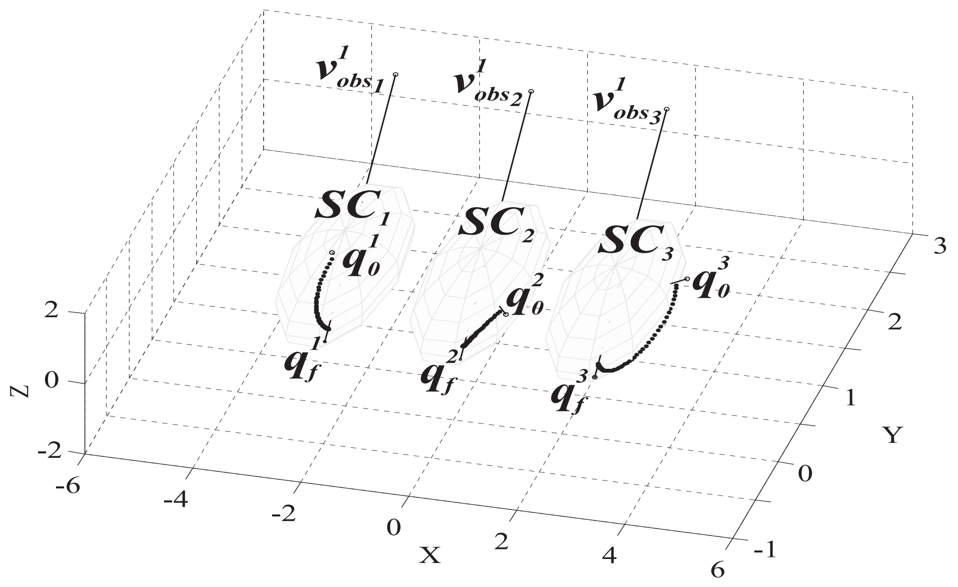


Figure 4.12: Reorientation to consensus attitude, with inter vehicle thruster plume avoidance.

The final consensus quaternion is $q_f = [-0.8167\ 0.4807\ -0.2396\ 0.2112]^T$ as shown in Figure 4.14. This corresponds to the normalized average quaternion of the initial attitude quaternions.

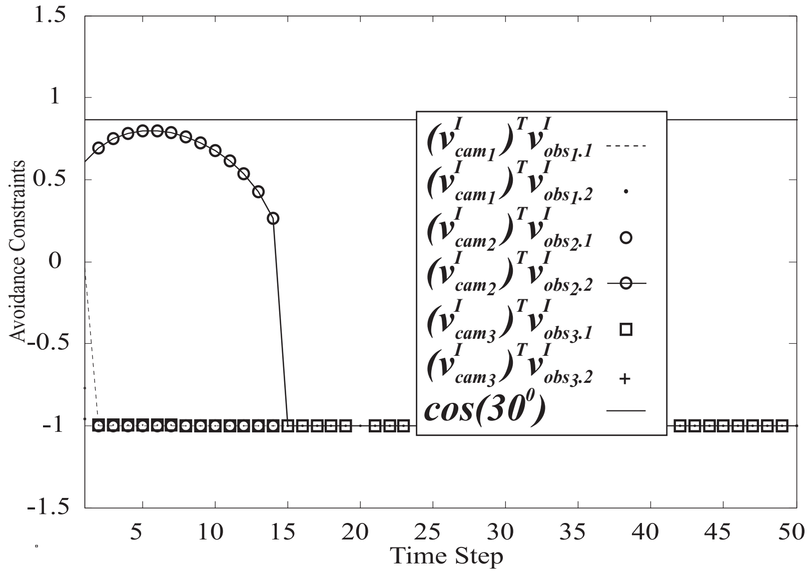


Figure 4.13: Avoidance constraints graph for Figure 4.12.

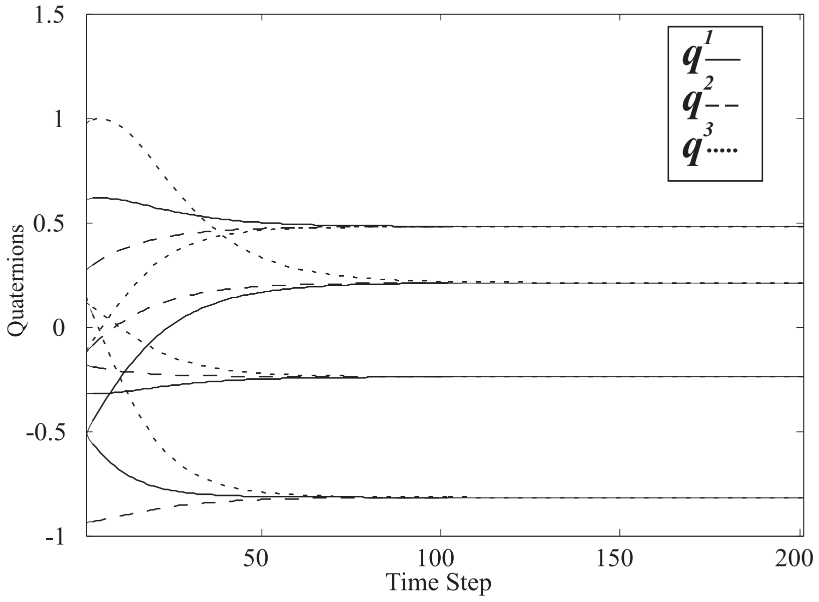


Figure 4.14: Attitude consensus graph for Figure 4.12.

4.5.6 Attitude Formation Acquisition with Avoidance

In this experiment SC_1, SC_2 and SC_3 will manoeuvre to consensus formation attitudes, with the sensitive instruments pointing 30° offsets from each other, about the z axis. The previous set of initial quaternions and coordinate frames in Section 4.5.5 were used, this time, relative offset quaternions for each spacecraft are included, where $q_1^{off} = [0\ 0\ 0\ 1]^T$, $q_2^{off} = [0\ 0\ 0.2588\ 0.9659]^T$, and $q_3^{off} = [0\ 0\ 0.5\ 0.866]^T$. It is desired that the sensitive instrument on each spacecraft avoid the thruster plumes emanating from the two other spacecraft by an angle of 30°, during the entire period of the manoeuvre. The trajectories are shown in Figure 4.15, and the avoidance graph in Figure 4.16 shows that no constraints are violated. The final consensus quaternions are $q_f^1 = [-0.6926\ 0.6468\ -0.2798\ 0.1541]^T$, $q_f^2 = [-0.8364\ 0.4455\ -0.2303\ 0.2212]^T$ and $q_f^3 = [-0.9232\ 0.2138\ -0.1652\ 0.2733]^T$ as shown in Figure 4.17. The quaternions are 30° apart about the same axis.

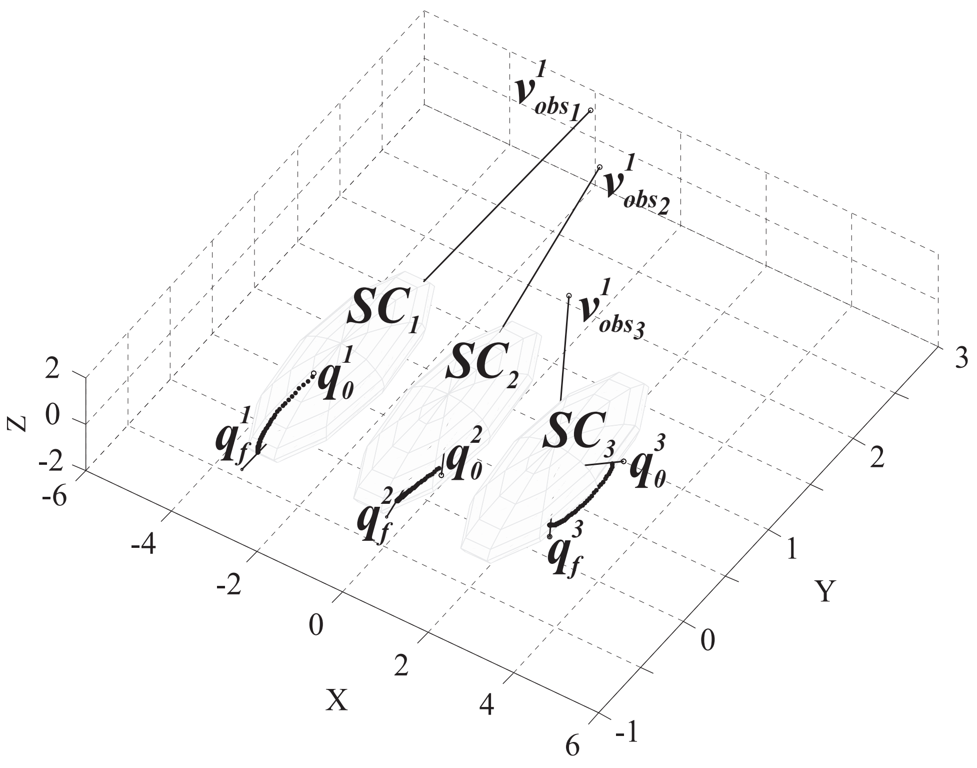


Figure 4.15: Reorientation to consensus formation attitude, with intervehicle thruster plume avoidance.

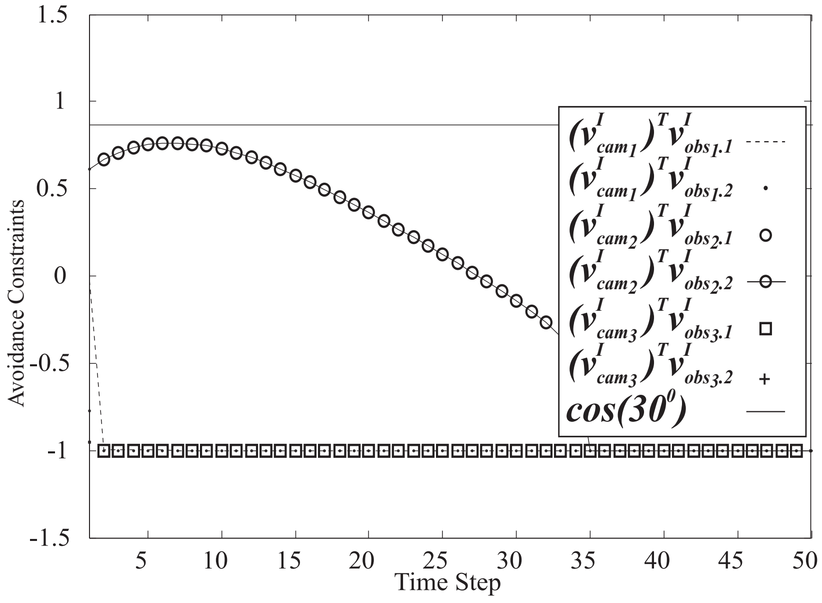


Figure 4.16: Avoidance constraints graph for Figure 4.15.

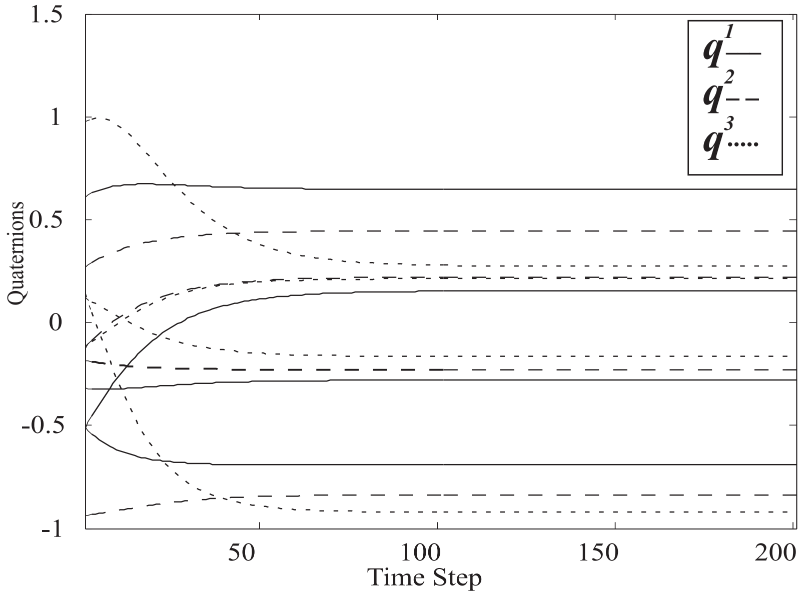


Figure 4.17: Attitude Consensus graph for Figure 4.15.

4.5.7 Computational Aspect of Current and Existing Approaches

In this section, a comparison is made between the results obtained using the approach developed in Section 4.4.1 also discussed by Okoloko and Kim (2010), the current proposed approach ($CURR$) developed in Sections 4.4.2-4.4.4 , and the one by Kim *et al.* (2010). As noted before, the main feature of the approach in Sections 4.4.2-4.4.4 is that the problem in question is decentralized, so that each spacecraft can solve its own small-sized optimization problem independently of states of neighbouring spacecraft. This feature has greatly helped to reduce computational complexity. The comparison measures used here are: (i) number of spacecraft (n); (ii) number of thrusters to avoid (n_t); (iii) number of decision variables (n_d); (iv) total number of constraints (m); (v) time required to solve the problem (t). We shall limit n to 2, because for more than two spacecraft the computational complexity and time required to solve the problem using the approach by Kim *et al.* (2010), becomes prohibitive. Also, the algorithm by Okoloko and Kim (2010) breaks down when more than two cone avoidance constraints are included. Since consensus was not implemented by Kim *et al.* (2010), we used the same experiment as in Section 4.5.4 above for the tests, with the number of thrusters n_t reduced to two for Okoloko and Kim (2010).

Table 4.1 summarizes the results. Although the work by Okoloko and Kim (2010) ran faster than the one of Kim *et al.* (2010) and $CURR$, this is partly due to the fact that n_t is one less for Okoloko and Kim (2010) than the other two approaches. It is clear that the computational complexity is greatly reduced in the current method Section 4.4.2-4.4.4. Moreover, the current tests were performed on a single hardware platform, but could be implemented on separate hardware platforms to significantly speed up the execution time of the current proposed approach. Thus, we can conclude that the approach in Sections 4.4.2-4.4.4 outperforms the ones by Kim *et al.* (2010) and Okoloko and Kim (2010). The same conclusion can be drawn by the fact that the compared three approaches are all LMI-based ones, and so their theoretical complexity is $O(n_d^2 m^{2.5} + m^{3.5})$ (Peaucelle *et al.*, 2002). Moreover, it should be noted that only the current proposed method ($CURR$) can be practically implemented for constrained attitude control of multiple spacecraft.

Table 4.1: Comparison of Current Approach With Kim *et al.* (2010) and Okoloko and Kim (2010)

$Approach$	n	n_t	n_d	m	t
Kim *et al.* (2010)	2	3	36	46	3h 26.29s
Okoloko and Kim (2010)	2	2	34	19	29s
$CURR$	2	3	13	12	1m 50.3s

4.6 Concluding Remarks

Some important issues are worth noting about the approaches presented in this chapter, which should be considered for practical implementations. First, when the norm constraint of (4.4.2) is included in the framework of $CURR$, the average quaternion obtained from Matlab simulation is erroneous. Although equation (4.4.2) is mathematically sound in theory, for numerical reasons this problem persists. Including this constraint directly in the consensus based Q-CAC framework is *only* necessary for the approach by Okoloko and Kim (2010), and not for $CURR$. If this constraint is removed and a normalization is programmatically calculated at each time step, an accurate average consensus value is obtained for $CURR$. Therefore for the approach $CURR$, normalization is realized programmatically every time step, without including the constraint (4.4.2). This normalized value is passed into the Q-CAC avoidance algorithm (4.4.13), which in turn returns normalized quaternions which are later passed back into the consensus framework for the next time step. This strategy has provided accurate results in the simulations, and the normalization also guarantees that the analysis and proofs presented still hold.

Chapter 5

Consensus and Optimization Based Collective Motion on a Sphere

---

This chapter presents a graph theoretic and optimization based algorithm for planning multiple collision free trajectories for a team communicating vehicles whose motions are constrained to evolve in a spherical coordinate system. The motion in a spherical coordinate system can be mapped into the equivalent motion on the surface of the unit sphere centered on the origin of the spherical coordinate system, by normalization of the position vectors. Like the algorithms presented in Chapters 3 and 4, consensus and Q-CAC is applied, this time to the problem of constrained motion on a sphere. However, one difference from the formulation in Chapters 3 and 4, is that this algorithm consists of a single-layer process in which a consensus optimization process and a Q-CAC based collision avoidance process run together in a single semidefinite program, defined as a set of LMI. Computer simulation results are presented for multiple vehicle position reconfiguration and formation control on a sphere, with dynamic collision avoidance (Okoloko, 2012*a*).

5.1 Introduction

In this chapter, a novel approach to constrained control of multiple vehicles navigating in a spherical coordinate system, is presented. The problem of control in a spherical coordinate system is a major aspect of navigation that has very practical applications to global and space navigation. Because the problem poses different challenges than the problem of control in 2D and 3D spaces it is important to consider it in a separate chapter of its own. The approach presented here is based on consensus theory and quadratically constrained attitude control (Q-CAC). Such algorithms have applications in planetary-scale mobile sensing networks in: air (Beard *et al.*, 2006); sea e.g. in remote-sensing and persistent sensing at ocean-basin scales (MBARI, 2006; Leonard *et al.*, 2007); space navigation and satellite cluster positioning (e.g. Mesbahi and Hadaegh, 2001; Blackwood *et al.*, 2002).

Most of the previous work on multi-vehicle motion planning has focused on two-dimensional (e.g. Justh and Krishnaprasad, 2004; Sepulchre *et al.*, 2007), and three-dimensional (e.g. Chandler *et al.*, 2001; Richards *et al.*, 2002; Kim *et al.*, 2004; Justh and Krishnaprasad, 2005; Scardovi *et al.*, 2005; Keviczky *et al.*, 2008; Okoloko and Basson, 2011) motion planning. Two-dimensional path planning is limited to the plane, while three-dimensional models are useful for planning motion control in volumetric 3D space. Both path planning models are limited when the motion is constrained to evolve on a sphere. The works of Paley (Paley, 2008; Hernandez and Paley, 2009) are beginning the research into the important area of distributed control on a sphere, and there is the need to explore the topic further.

Motivated by planetary-scale in-situ sensing networks such as the Argo Profiling Float Project (NOAA[1]), and the successful crossing of the Gulf Stream by an underwater glider (Nevala, 2005), Paley (2008) studied motion planning algorithms for a system of self-propelled particles travelling on the surface of a sphere. He developed a model of self-propelled particles that move at constant speed on the surface of a sphere, using a Lie group representation to identify circular formations of steady motions of the particles around a fixed, small circle on the sphere, as relative equilibria of the model. He also provided mathematically justified shape control laws that stabilize the set of circular formations, assum-

[1] See http://floats.pmel.noaa.gov/

ing either time-invariant and undirected or time-varying and directed particle interaction. He also proposed a shape control to isolate circular formations of particles with symmetric spacing, by using Laplacian control. This work has also been extended to stabilization of collective motion on a rotating sphere (Hernandez and Paley, 2009).

The work of Paley (2008) is based on the works of Justh and Krishnaprasad (2004, 2005), where a geometric approach to the gyroscopic control of vehicle motion in planar and three-dimensional particle models was developed, for formation acquisition and control with collision avoidance, in free space. They found that for their unconstrained gyroscopic control system on $SE(3)$, there are three possible types of relative equilibria: (i) parallel motion with arbitrary spacing; (ii) circular motion with a common radius, axis of rotation, direction of rotation, and arbitrary along-axis spacing; and (iii) helical motion with a common radius, axis of rotation, direction of rotation, along-axis speed (pitch), and arbitrary along-axis spacing.

The control system developed by Paley (2008) conforms to the number (ii) type of relative equilibrium described above. That is, the control system is capable of circular motion of particles, with a common radius, axis of rotation, and direction of rotation. This means that, at steady state, all of the particles converge to a circular pattern on the sphere, while moving in the same direction.

The geometric approach to the gyroscopic control of vehicle motion developed by Justh and Krishnaprasad (2004, 2005) is effective in formation control of multiple systems in unconstrained spaces, and for formations that conform only to the relative equilibria described above. However, the approach cannot be applied to the more general formation control problem involving: (i) constrained spaces which contain static obstacles such as clutter; (ii) constrained vehicle motion; and (iii) arbitrary formations which are different from the three relative equilibria described above. This motivates the development of a new approach which is presented in this chapter.

In the work of Paley (2008), particle dynamics was derived using a spherical coordinate system consisting of the azimuth angle θ_k, the polar angle ϕ_k, and the (fixed) radius ρ_0 from the center of the unit sphere, centered at the origin $\mathbf{0}$. An inertia reference frame centered on **0** is defined, and in order to specify the kinematics of each particle, three additional reference frames are introduced. The dynamics and kinematics of each particle are defined by Euler rotations about the three coordinate frames.

In this work, a new approach is presented to the general problem of constrained path planning on the sphere with avoidance of collisions, using consensus and quadratically constrained attitude control (Q-CAC) optimization theory, earlier described in Chapters 3 and 4. The Laplacian matrix of the communication graph **L** is used in a semidefinite program to plan consensus trajectories on the sphere, and the concept of Q-CAC is used to incorporate collision avoidance by maintaining specified minimum relative angles between vehicles. The difference between the approach presented here and the geometric approach (Justh and Krishnaprasad, 2004; Paley, 2008; Hernandez and Paley, 2009) are: (i) the algorithm can be used for motion control in both constrained and unconstrained spaces on the sphere, e.g. planning consensus trajectories around static obstacles on the sphere; (ii) the geometric approach is for unconstrained vehicle motion, while the approach presented here can be applied to constrained vehicle motion; (iii) different kinds of formations are possible on the sphere, including circular formations; (iv) the minimum time trajectory can be achieved by using the approach developed in this work.

The rest of the chapter is organized as follows. The problem statement is presented in Section 5.2 and a mathematical background is in Section 5.3. The solution and convergence analysis is in Section 5.4. Simulation results are in Section 5.5 and concluding remarks follows in Section 5.6.

5.2 Problem Statement

Given a set of communicating vehicles whose motions are constrained to evolve in a sphere. The vehicles are randomly positioned in the sphere, with reference to a coordinate frame centred on the centroid of the sphere. Given the initial states $x^i(t_0) \in \mathbb{R}^3,\ i = 1, \cdots, n$, a set of obstacles $x^j_{obs} \in \mathbb{R}^3,\ j = 1, \cdots, m$, and the Laplacian matrix of their communication graph **L**, our concern is to drive $x^i(t_0)$ to a consensus position $x^c = x(t_f)$, or to a consensus formation, while satisfying collision avoidance constraints. Note that $x(t_f)$ need not a priori be known to any of the vehicles.

To better understand the problem, consider Figure 5.1. For the purpose of developing a solution we shall normalize the respective position and obstacle vectors, so that the problem is reduced to constrained control on the unit sphere centered on **0**. Therefore x^i and x^i_{obs} are considered as unit vectors. The angle between vehicles i and j is θ^{ij}, and the angle between vehicle i and obstacle k is φ^{ik}. The motion control problem is to drive all x^i to a consensus position or to a formation in the sphere, while avoiding each other and also avoiding the x^i_{obs} along the way. After obtaining the solution trajectories as unit vectors, the actual desired vehicle trajectories are recovered from the normalized unit vectors via scalar multiplication. Note that in this development, a vehicle is modeled as a point mass.

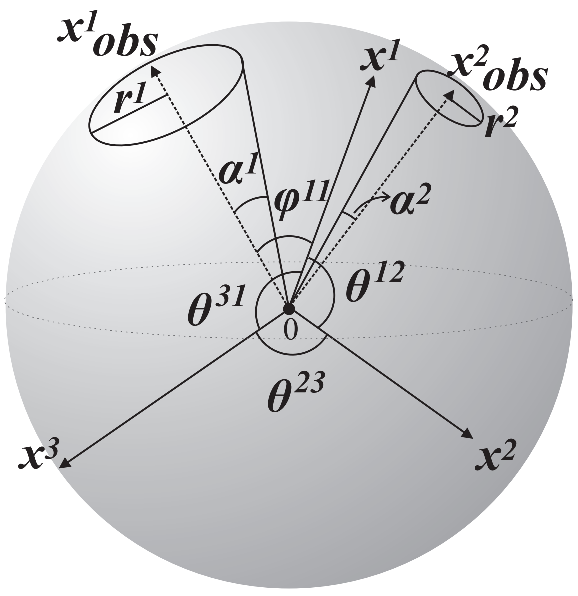


Figure 5.1: Constrained position control on a sphere.

Observe that this path planning problem is similar to the attitude control problem presented in Chapter 4. This similarity comes from the view point that the motion of the vehicles in the sphere can be deduced from the rotations of the respective unit vectors originating from the centre of the sphere, and corresponding to the positions of the vehicles on the sphere. Essentially, any kind of motion about a spherical coordinate system is a rotation of some vector by some angle, about some axis, and the angle and axis defines a quaternion vector[2].

Like the attitude control problem, the problem stated above has two major parts: *consensus* and *collision avoidance*. The consensus part is that of driving the positions of the vehicles to a consensus position in the sphere. If no relative spacing is specified, the consensus position is usually the centroid of the initial positions, meaning that the vehicles should eventually rendezvous to a single point on the sphere in finite time. We have seen a similar example in the attitude rendezvous problem in single coordinate frame, however, the position rendezvous is different, even though the general formulations are similar. To obtain formation configurations, relative offset vectors cannot be applied here, as it was applied for position formations in Chapter 3 and attitude formations in Chapter 4. Rather, formations are realized by defining and maintaining minimum angular separations between the vehicles, using Q-CAC.

The problem of collision avoidance is resolved by applying Q-CAC, which has been described in Chapter 3 where it was applied to motion control in 2D and 3D, and Chapter 4 where it was applied in attitude control. The method of applying Q-CAC for collective motion control in a sphere is similar to the ones previously described. However the main difference is that the consensus problem and the Q-CAC problem are solved together in a single semidefinite program, and quaternions are not involved in the collision avoidance process.

5.3 Mathematical Background

The relevant mathematical background for this work are in Sections 3.2, 3.4, 4.3 and 4.4, and the appropriate section will be referred to whenever information contained therein is required.

[2]This is indeed obvious from the work of Paley (2008) where particle dynamics is derived from Euler rotations.

5.4 Solution

In this section, a solution is developed to the problem stated in Section 5.2. The solution involves four intermediate steps: (i) synthesis of position consensus on a sphere; (ii) formulation of Q-CAC based collision avoidance; (iii) formulation of Q-CAC based formation control; (iv) formulation of collision free arbitrary reconfigurations on a sphere.

5.4.1 Synthesis of Position Consensus on a Sphere

The problem under consideration can be coded as a set of LMI, and solved for consensus position trajectories on the sphere, using available optimization software tools such as Sedumi (Sturm, 1998) and Yalmip (Lofberg, 2004). To this end, a new objective function $(5.4.1)$ is defined, and augmented with an arbitrary number of constraints which are defined as we proceed.

Given the set of initial unit position vectors in a column vector $\mathbf{x}(t_0)$, and the associated graph Laplacian $\mathbf{L}$, we want to synthesize a Laplacian-like stochastic matrix $\mathfrak{L}$ that drives $\mathbf{x}$ to consensus, while maintaining a unit norm constraint. This matrix is similar to the matrix $\mathbf{P}$ in (4.3.6), which was synthesized for the attitude control problem in Chapter 4. Therefore the development is essentially the same as the attitude control algorithm, but the dimension of the state matrix $\mathfrak{L}$ is less than the dimension of the state matrix $\mathbf{P}$ by n, where n is the number of vehicles. This reduction is due to three dimensions for unit vectors, rather than four for quaternion vectors.

To obtain $\mathfrak{L}$, we begin by defining n positive semidefinite matrix variables, $\Lambda^1, \Lambda^2, \cdots, \Lambda^n$, for n vehicles, $\Lambda^i \in \mathcal{S}^3$, where $\mathcal{S}^n$ denotes the set of $n \times n$ positive semidefinite matrices. Then

$$\mathfrak{L}(t) = \underbrace{\begin{bmatrix} \Lambda^1(t) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \Lambda^n(t) \end{bmatrix}}_{\Lambda(t)} \underbrace{\begin{bmatrix} l_{11}\mathbf{I}_3 & \dots & l_{1n}\mathbf{I}_3 \\ \vdots & \ddots & \vdots \\ l_{n1}\mathbf{I}_3 & \dots & l_{nn}\mathbf{I}_3 \end{bmatrix}}_{\Gamma=\mathbf{L}\otimes\mathbf{I}_3},$$

where l_{ij} are elements of $\mathbf{L}$.

The new collective objective function is therefore given as

$$\dot{\mathbf{x}}(t) = -\mathfrak{L}(t)\mathbf{x}(t). \tag{5.4.1}$$

If any vehicle i can communicate with y other vehicles, then the individual consensus protocol for i is given as

$$\dot{x}^i(t) = -\left[y\Lambda_1^i(t) \quad -\Lambda_2^i(t) \cdots - \Lambda_y^i(t)\right]\left[x_1^T(t) \;\; x_2^T(t) \cdots x_y^T(t)\right]^T$$

$$= -\,\mathfrak{L}^i(t)\left[x_1^T(t) \;\; x_2^T(t) \cdots x_y^T(t)\right]^T,$$

with Euler's first order discrete time equivalent

$$x_{k+1}^i = x_k^i - \Delta t \dot{x}^i. \tag{5.4.2}$$

To include norm constraints we follow (4.4.2),

$$(x^i)_k^T(x_{k+1}^i - x_k^i) = 0. \tag{5.4.3}$$

Equations (5.4.2) and (5.4.3) are unit Cartesian vector equivalents of (4.4.1) and (4.4.2). Combining (5.4.2) and (5.4.3), we have a system that drives the positions $x^i(0)$ to consensus on a sphere.

Theorem 5.1 *The time varying system* (5.4.1) *achieves consensus assuming connectivity of* **L**.

Proof: The proof follows directly from the proof of Theorem 4.1.

5.4.2 Formulation of Q-CAC based Collision Avoidance

Next, to incorporate collision avoidance, we follow the same approach to collision avoidance for multiple vehicles attitude reconfiguration in the same coordinate frame, centred on the unit sphere (Okoloko and Kim, 2010). The difference however is that we want to control positions, not attitudes, and for this it is sufficient to control the corresponding angles between the vehicles and obstacles vectors. Therefore, attitude constraint (3.4.4) remains valid for this problem, and thus the solution (3.4.9) can be applied here. As an example, consider Figure 5.1. Suppose we want the time evolution of the position vectors $x^1(t)$, $x^2(t)$ and $x^3(t)$ to avoid two constraint regions around x^1_{obs} and x^2_{obs}, defined by cones, whose base radii are r^1 and r^2, respectively. Let the angle between vehicles i and j be θ^{ij}, and let the angle between vehicle i and obstacle k be φ^{ik}. Then the requirement for collision avoidance are

$$\varphi^{11}(t) \geq \alpha^1,\ \varphi^{21}(t) \geq \alpha^1,\ \varphi^{31}(t) \geq \alpha^1,\ \varphi^{12}(t) \geq \alpha^2,\ \varphi^{22}(t) \geq \alpha^2,\ \varphi^{32}(t) \geq \alpha^2,$$

$\forall\ t \in [t_0, t_f]$. The equivalent constraints are

$$\begin{aligned} x^1(t)^T x^1_{obs} &\leq \cos\alpha^1, \\ x^2(t)^T x^1_{obs} &\leq \cos\alpha^1, \\ x^3(t)^T x^1_{obs} &\leq \cos\alpha^1, \\ x^1(t)^T x^2_{obs} &\leq \cos\alpha^2, \\ x^2(t)^T x^2_{obs} &\leq \cos\alpha^2, \\ x^3(t)^T x^2_{obs} &\leq \cos\alpha^2. \end{aligned}$$

For this problem it is sufficient to include the following LMI avoidance constraints:

$$\begin{bmatrix} 2(\mu + \cos\alpha^1) & \begin{bmatrix} x^1(k+2) \\ x^1_{obs} \end{bmatrix}^T \\ \begin{bmatrix} x^1(k+2) \\ x^1_{obs} \end{bmatrix} & \mathbf{M} \end{bmatrix} \geq 0, \tag{5.4.4}$$

$$\begin{bmatrix} 2(\mu + \cos\alpha^1) & \begin{bmatrix} x^2(k+2) \\ x^1_{obs} \end{bmatrix}^T \\ \begin{bmatrix} x^2(k+2) \\ x^1_{obs} \end{bmatrix} & \mathbf{M} \end{bmatrix} \geq 0, \tag{5.4.5}$$

$$\begin{bmatrix} 2(\mu+\cos\alpha^1) & \begin{bmatrix} x^3(k+2) \\ x^1_{obs} \end{bmatrix}^T \\ \begin{bmatrix} x^3(k+2) \\ x^1_{obs} \end{bmatrix} & \mathbf{M} \end{bmatrix} \geq 0, \tag{5.4.6}$$

$$\begin{bmatrix} 2(\mu+\cos\alpha^2) & \begin{bmatrix} x^1(k+2) \\ x^2_{obs} \end{bmatrix}^T \\ \begin{bmatrix} x^1(k+2) \\ x^2_{obs} \end{bmatrix} & \mathbf{M} \end{bmatrix} \geq 0, \tag{5.4.7}$$

$$\begin{bmatrix} 2(\mu+\cos\alpha^2) & \begin{bmatrix} x^2(k+2) \\ x^2_{obs} \end{bmatrix}^T \\ \begin{bmatrix} x^2(k+2) \\ x^2_{obs} \end{bmatrix} & \mathbf{M} \end{bmatrix} \geq 0, \tag{5.4.8}$$

$$\begin{bmatrix} 2(\mu+\cos\alpha^2) & \begin{bmatrix} x^3(k+2) \\ x^2_{obs} \end{bmatrix}^T \\ \begin{bmatrix} x^3(k+2) \\ x^2_{obs} \end{bmatrix} & \mathbf{M} \end{bmatrix} \geq 0, \tag{5.4.9}$$

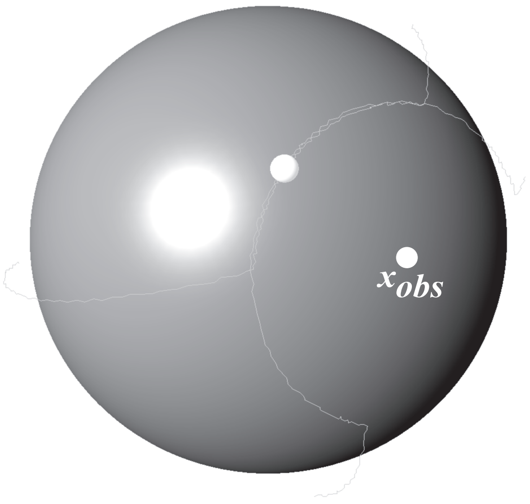
where the matrix **M** is defined in p. 46. Figure 5.2 shows the result of applying the above strategy to the rendezvous of four vehicles on the unit sphere of 1 km radius, with avoidance of a static obstacle x_{obs}. In this experiment $\alpha = 30°$.

5.4.3 Formulation of Q-CAC based Formation Control

To obtain formation patterns, relative spacing is defined between individual vehicles using the method presented above, by specifying a minimum angular separation β^{ij} between any two vehicles i and j. The set of avoidance constraints that will result in the formation pattern is then defined as $\theta^{ij} \geq \beta^{ij}\ \forall\, i, j$. To include intervehicle collision avoidance for n vehicles, the avoidance requirements result in $P(n-2) = \frac{n!}{(n-2)!}$ extra constraints, which are included along with the static obstacle avoidance constraints such as (5.4.4)-(5.4.9). Figure 5.3 shows the result of applying the above strategy to the rendezvous with intervehicle avoidance and static obstacle avoidance, of four vehicles, using the fully connected Topology 2 in Figure 5.7. In this experiment $\alpha = 30°$, and the minimum angular separations between the vehicles is set at a constant value $\beta^{ij} = 20° \ \forall \ i, j$. Therefore, in addition to four static obstacle avoidance constraints (such as (5.4.4)-(5.4.7) with $\alpha^1 = \alpha$), twelve intervehicle collision avoidance constraints are included as

$$\begin{bmatrix} 2(\mu + \cos\beta^{ij}) & \begin{bmatrix} x^i(k+2) \\ x^j(k+2) \end{bmatrix}^T \\ \begin{bmatrix} x^i(k+2) \\ x^j(k+2) \end{bmatrix} & \mathbf{M} \end{bmatrix} \geq 0, \tag{5.4.10}$$

$\forall i, j \ (i \neq j)$.



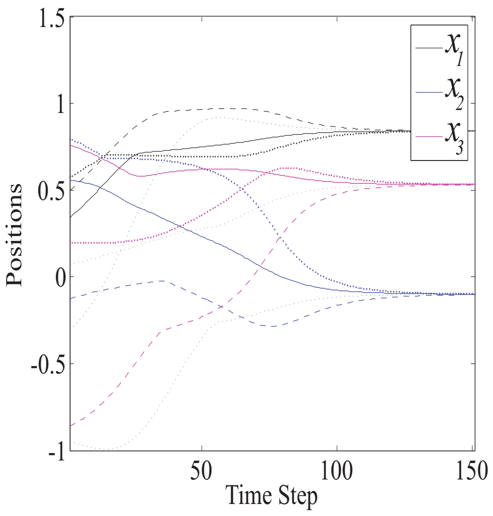


Figure 5.2: Four-vehicle rendezvous on a unit sphere with collision avoidance of a static obstacle.

5.4.4 Formulation of Collision free Arbitrary Reconfigurations on a Sphere

Consider a more traditional reconfiguration problem in which several vehicles have to change their positions to any desired final positions. If the final positions are known, the problem can be formulated as a consensus problem involving multiple vehicles with an equal number of static virtual leaders.

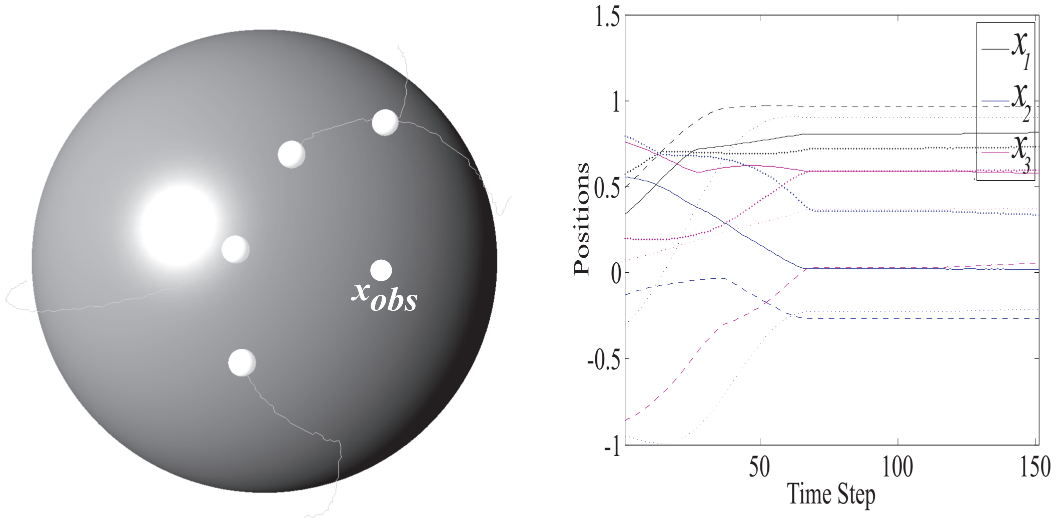


Figure 5.3: Four-vehicle formation acquisition on a unit sphere with collision avoidance of a static obstacle, and with intervehicle collision avoidance.

The virtual leaders are located at the known desired final positions, and each vehicle is connected to its corresponding virtual leader via a leader-follower digraph. An example topology for three vehicles is shown in Figure 5.4. In the figure, the vertices in dashed circles are the virtual leaders states, while the vertices with solid circles correspond to the real vehicles states. There are three leader follower digraphs (with pointing arrows) which do not commute, and there is an undirected graph which enables the vehicles to communicate. This graph provides intervehicle communication which is used to detect and avoid potential collisions.

Let the state of a virtual leader vehicle i be $x_v^i(t)$, then the corresponding leader-follower Laplacian matrix for each vehicle pair $\mathbf{x}^i(t) = [x_v^i(t)^T \;\; x^i(t)^T]^T$ is

$$\mathbf{L}^i(t) = \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix},$$

and the collective dynamics of $\mathbf{x}^i(t)$ is

$$\dot{\mathbf{x}}^i(t) = -\begin{bmatrix} \Lambda^i(t) & \mathbf{0} \\ \mathbf{0} & \Lambda^i_v(t) \end{bmatrix} \left(\mathbf{L}^i(t) \otimes \mathbf{I}_3\right) \mathbf{x}^i(t).$$

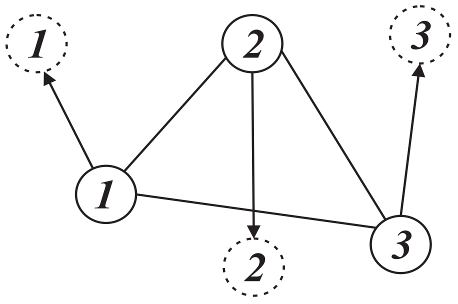


Figure 5.4: Multiple virtual leaders graph topology with an undirected topology.

Figure 5.5 shows the result for applying this strategy to the reconfiguration of two vehicles exchanging their positions with collision avoidance.

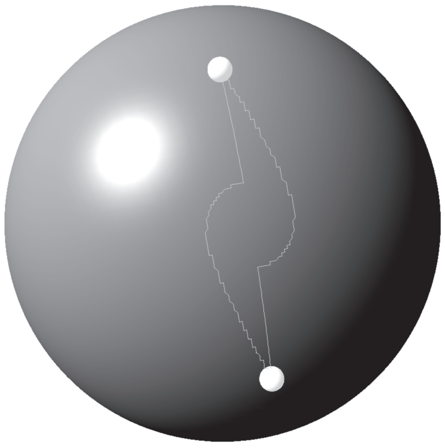


Figure 5.5: Reconfiguration of two vehicles on a unit sphere, with collision avoidance.

5.5 Practical Application to Air Traffic Control

Consider the problem of *separation* in air traffic control. This is the concept of keeping an aircraft outside a minimum distance from all other aircraft in order to reduce the risk of those aircraft colliding, as well as prevent accidents. There are two aspects of separation: *vertical separation* in which aircraft are required to maintain a standard minimum distance apart in altitude; *horizontal separation* in which aircraft are required to maintain a standard minimum lateral or longitudinal distance apart. Standard criteria for minimum horizontal and vertical separation distances are used across the world in air traffic control (Kuchar and Yang, 2000).

We shall consider how to apply the algorithm developed in this chapter to the problem of horizontal separation, by reducing the problem to that of constrained control on a unit sphere.

Consider Figure 5.6[3]. Two aircraft i and j, flying at altitudes of $h^i(t)$ and $h^j(t)$ at time t, are required to maintain a desired minimum horizontal separation of d_d^{ij} from each other at all times. Given d_d^{ij}, the desired angular separation β^{ij} can be computed. The aircraft altitudes, relative to the center of the Earth, are $z^i = h^i + r_e$ and $z^j = h^j + r_e$. Here r_e is the radius of the Earth. The current position of any aircraft i can be uniquely determined relative to the Earth coordinate frame by using its current latitude, longitude and altitude. Given that the current positions are measured as $x^i(t)$ and $x^j(t)$ in Earth coordinate frame, the actual distance $d_a^{ij}(t)$ between i and j at time t can be uniquely computed. It is desired that $d_a^{ij}(t) \geq d_d^{ij}$ for all time t.

To apply the algorithm, we need to determine the current actual angular separation $\theta^{ij}(t)$ between i and j. One way to obtain $\theta^{ij}(t)$ is to normalize $x^i(t)$ and $x^j(t)$. Let $\hat{x}^i(t)$ and $\hat{x}^j(t)$ be the corresponding normalized vectors of $x^i(t)$ and $x^j(t)$, then $\theta^{ij}(t) = \cos^{-1}(\hat{x}^i(t)^T \hat{x}^j(t))$. Alternatively, it is easy to compute $\theta^{ij}(t)$ using the cosine rule as long as $x^i(t)$ and $x^j(t)$ are known.

Following the normalization approach, the horizontal separation constraint is transformed to the constraint $\theta^{ij}(t) \geq \beta^{ij}$. Thus the horizontal separation problem is equivalent to the problem of constrained control on the unit sphere of 1km radius, where the constraint between i and j is $\cos^{-1}(\hat{x}^i(t)^T \hat{x}^j(t)) \geq \beta^{ij}$ or $\hat{x}^i(t)^T \hat{x}^j(t) \leq \cos(\beta^{ij})$.

However, the aircraft are not navigating on the unit sphere, and may not necessarily

[3]All units are measured in kilometres.

be flying at the same altitude. Therefore to recover the desired avoidance trajectory for i and j, after successfully computing the avoidance trajectories in unit vector form $\hat{x}^i(k+2)$ and $\hat{x}^j(k+2)$, the actual desired avoidance trajectories are computed as $x^i(k+2) = z^i(k)\hat{x}^i(k+2)$ and $x^j(k+2) = z^j(k)\hat{x}^j(k+2)$.

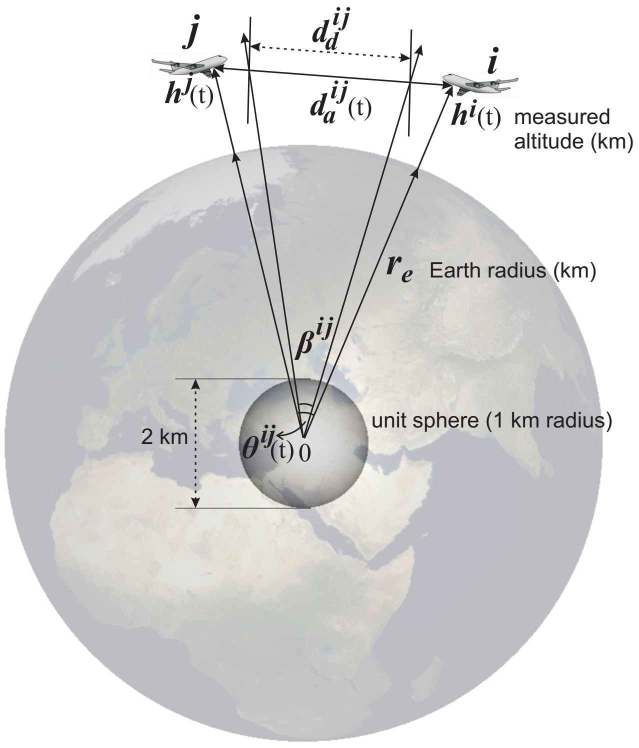


Figure 5.6: Aircraft horizontal separation problem.

The algorithm effectively solves the problem of horizontal separation, irrespective of a difference between $h^i(t)$ and $h^j(t)$. It is also straightforward to apply this strategy to maritime vessel traffic control, and to satellite cluster positioning.

5.6 Simulation Results

In this section, three simulation results are presented for coordinated control on the unit sphere. The first experiment is to test rendezvous on the sphere, i.e. convergence without avoidance constraints. The second experiment is for formation acquisition on the sphere with collision avoidance. The third experiment is to test arbitrary reconfigurations on the sphere with collision avoidance. Three different topologies, shown in Figure 5.7, are used to solve different problems in the experiments. In the figure, Topology 2 (left) is a fully connected communication graph with no leader, Topology 3 (center) is a cyclic communication graph with one leader, node 1, and Topology 4 (right) is a cyclic communication graph with no leader.

Note that we have adopted the unit sphere of 1 km radius for the simulations because, as shown earlier, the problem of navigation in a spherical coordinate system can be solved by control on the unit sphere. Moreover, the results are easier to visualize on the unit sphere. Therefore, the unit of measurements are in kilometres. The results presented here can be directly applied to horizontal separation of aircraft, watercraft and satellites, simply by transforming actual positions into the unit sphere, solving to obtain the solution trajectories, and transforming the solutions back to actual desired trajectories.

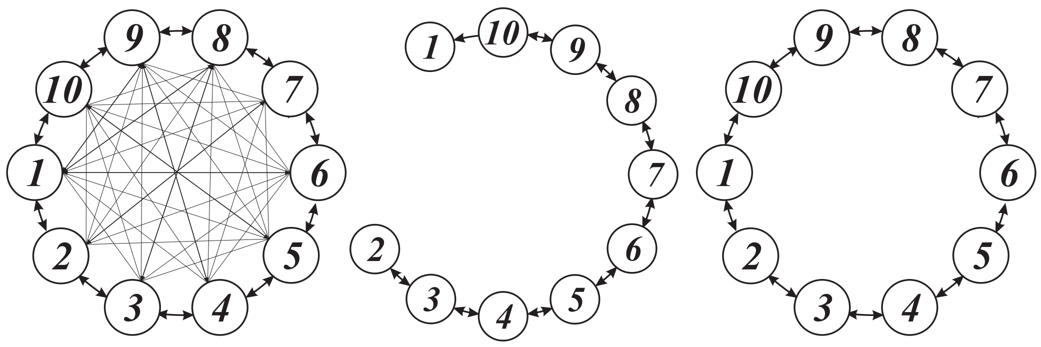


Figure 5.7: Topology 2 (left), Topology 3 (center) and Topology 4 (right).

Sedumi (Sturm, 1998) and Yalmip (Lofberg, 2004) were used for solving all the optimization problems in this section. The simulations were done with Matlab R2009a on an Intel Core(TM)2 Duo P8600 @ 2.40GHz with 2 GB RAM, running Windows 7.

5.6.1 Rendezvous on a Sphere without Avoidance

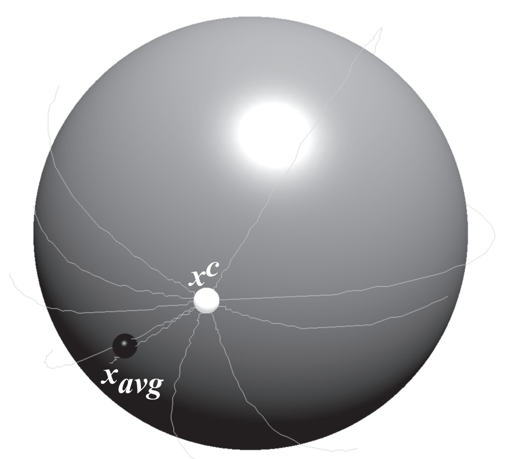
In this experiment, ten vehicles will converge to a consensus position on the sphere, using Topology 2. The initial positions are:

$$\begin{aligned}
x^1(0) =& [0.3417\ 0.5555\ 0.7581]^T,\\
x^2(0) =& [0.496\ -0.127\ -0.8589]^T,\\
x^3(0) =& [-0.3045\ -0.9497\ 0.073]^T,\\
x^4(0) =& [0.5735\ 0.7952\ 0.1967]^T,\\
x^5(0) =& [-0.8005\ -0.3867\ -0.458]^T,\\
x^6(0) =& [-0.3727\ -0.7372\ 0.5637]^T,\\
x^7(0) =& [0.0355\ -0.5117\ -0.8585]^T,\\
x^8(0) =& [-0.6553\ -0.7428\ -0.1371]^T,\\
x^9(0) =& [0.9188\ -0.2446\ -0.3094]^T,\\
x^{10}(0) =& [-0.0261\ -0.8773\ -0.4792]^T.
\end{aligned}$$

The final consensus position obtained from this experiment is $x^c = [0.3818\ -0.8794\ -0.2845]^T$, which is significantly different from the normalized average of the initial positions $x_{avg} = [0.0579\ -0.9042\ -0.4231]^T$. This error results from the normalization procedure of (5.4.3). The results are presented in Figure 5.8.

To circumvent the error problem, one can create a virtual leader vehicle that corresponds to the average of the initial positions, or to any desired point of rendezvous. Then the virtual leader vehicle is used as the single leader in a leader-follower architecture. Figure 5.9 (left) shows the result for synthesizing a virtual leader vehicle whose initial position is x_{avg}, the virtual leader is then used with Topology 2. The figure on the right was obtained by using x^1 as the leader vehicle, with Topology 3.

The strategy demonstrated in this simulation can be practically applied to rendezvous of multiple UAVs and to satellite cluster positioning, provided they are flying at different altitudes.



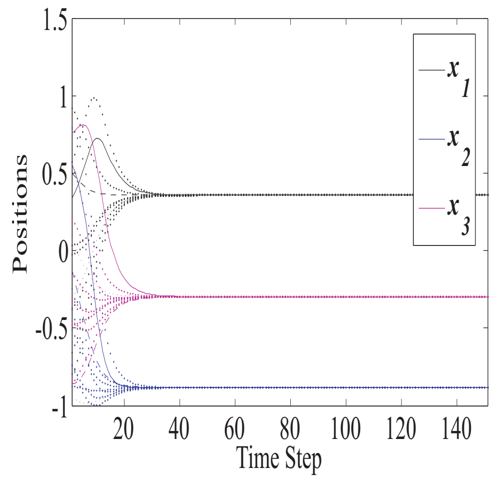
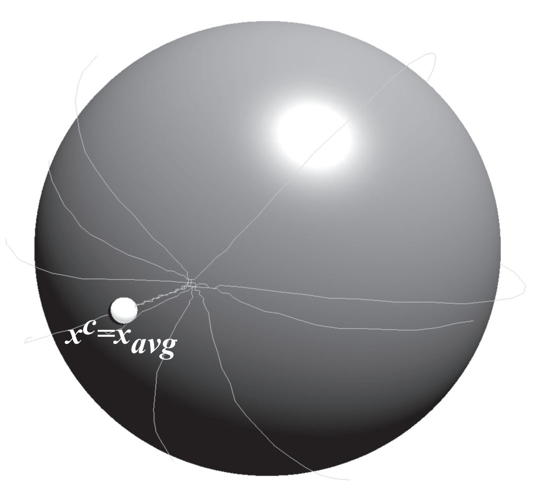


Figure 5.8: Ten-vehicle rendezvous on a sphere without avoidance constraints.



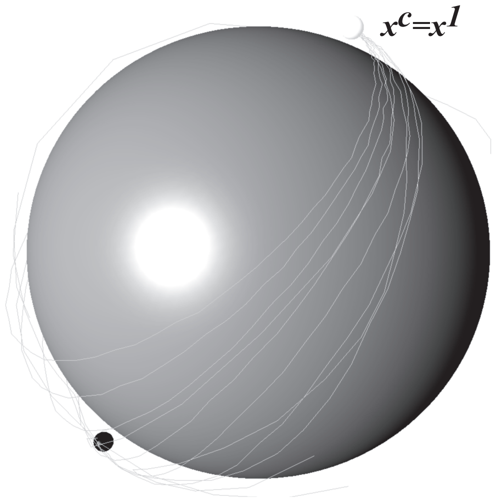


Figure 5.9: Ten-vehicle rendezvous on a sphere without avoidance constraints using a leader-follower topology, where x_{avg} is the leader (left), and x^1 is the leader (right).

5.6.2 Formation Acquisition on a Sphere with Avoidance

The first example of formation acquisition was presented in Figure 5.3, for four vehicles. In this simulation, ten vehicles will converge to a formation on the sphere. To realize the formation, they should maintain a relative spacing with one another, while also avoiding a static obstacle. For the static obstacle avoidance, $\alpha = 30°$, and to maintain the relative spacing between the vehicles, $\beta^{ij} = 20° \; \forall \; i, j = 1, \cdots, 10, \; i \neq j$. The initial positions are the same as those used in the previous experiment. The result for Topology 2 is shown in Figure 5.10 (left), while Figure 5.10 (right) shows the result obtained using Topology 4.

Topology 4 is a cyclic graph which produces a circulant **L**, whose dynamics lead to swirling motion. If no relative spacing is specified for the vehicles, the circular motion converges to a point. When relative spacing are specified for each of the vehicles, the motion obtained from this Laplacian corresponds to the relative equilibrium type (ii), which was presented in Section 5.1. This is similar to the result obtained by Paley (2008). Using a circulant matrix such as that of Topology 4, one is able to vary the radius of the circular formation achieved since $r = \cos\theta^{ij}$. This is done by setting θ^{ij} equal for all i, j and varying its size with time. If the magnitude of θ is reduced, the radius of formation structure obtained also reduces, and vice versa. Figure 5.11 (left) shows the result for setting $\theta^{ij} = 30° \;\forall\; i, j$ for four vehicles. The center and right figures show the results for ten vehicles, as θ^{ij} moves gradually from 20° towards 0°. Indeed if $\theta^{ij} = 0 \;\forall\; i, j$, then the vehicles will rendezvous to a point.

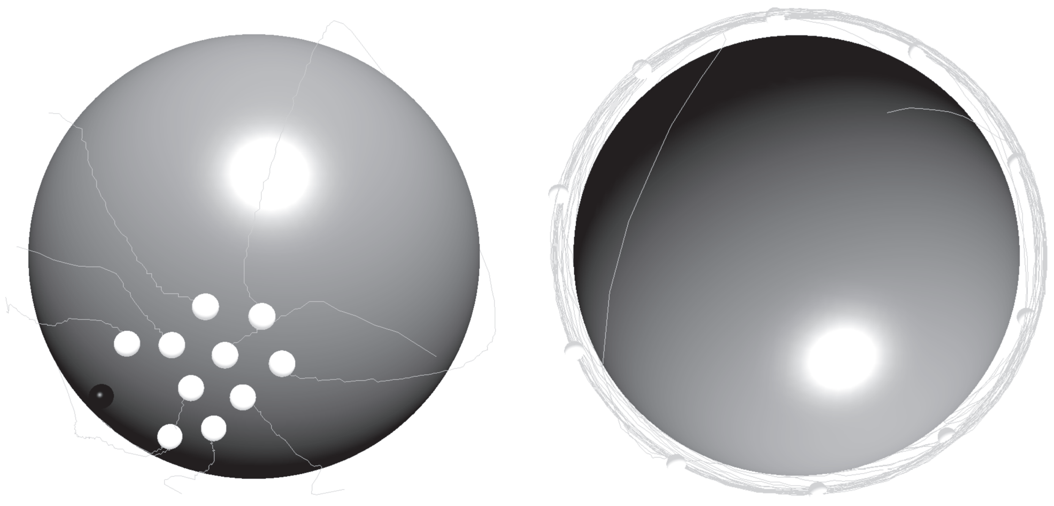


Figure 5.10: Ten-vehicle formation acquisition using Topology 2 (left), and using Topology 4 (right).

Using a fully connected graph such as Topology 2, the result is different. The vehicles converged to an irregular formation. Once the formation has been realized, the rigid body of the formation can be controlled in many ways.

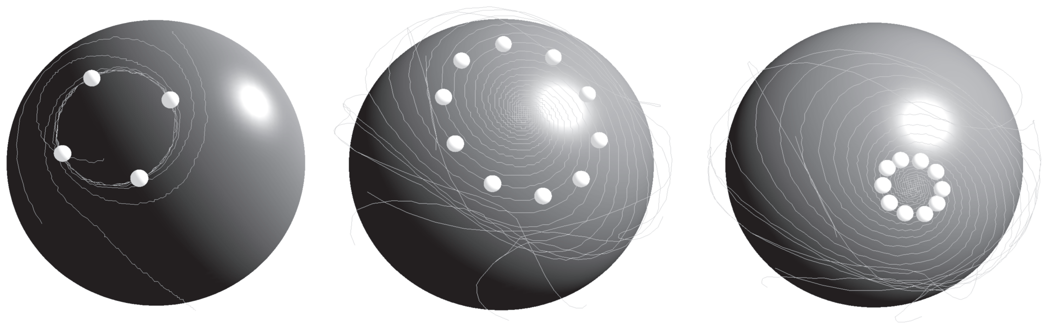


Figure 5.11: Four-vehicle formation acquisition using Topology 4 with $\beta^{ij} = 30°$ (left), and ten-vehicle formation acquisition, using Topology 4, with β^{ij} moving from 20° to 0° (center and right).

For example, the rigid body as a virtual structure can be made to rotate with constant velocity around any axis of choice on the sphere, by choosing the appropriate equivalent quaternion as the plant matrix. Therefore, for a chosen quaternion q, the discrete time dynamics of the formation is given as

$$\mathbf{x}(k+1) = \mathfrak{R}(k)\mathbf{x}(k), \tag{5.6.1}$$

where $\mathfrak{R} = \mathbf{I}_n \otimes \mathbf{R}$, and

$$\mathbf{R} = ((2q_4)^2 - 1)\mathbf{I}_3 + 2\bar{q}\bar{q}^T - 2q_4\bar{q}^\times,$$

is the rotation matrix corresponding to the attitude quaternion q, defined in Section 4.4.2, $\mathbf{I}_n$ is an $n \times n$ identity matrix, and n is the number of vehicles. Figure 5.12 (left) shows the result for rotating the rigid body formation of Figure 5.10 (left) constantly about the z axis by $1°$ angular rotation, the corresponding quaternion being $q = [0\ \ 0\ \ 0.0087\ \ 0.99]^T$. Figure 5.12 (right) shows the result for rotating the rigid body about the centroid of initial positions (x_{avg}) constantly by $1°$, the corresponding quaternion being $q = [0.0005\ \ -0.0079\ \ -0.0037\ 0.99]^T$.

The strategy demonstrated in this simulation can be practically applied to the separation of multiple aircraft, satellite cluster formation control, and formation control of multiple watercraft.

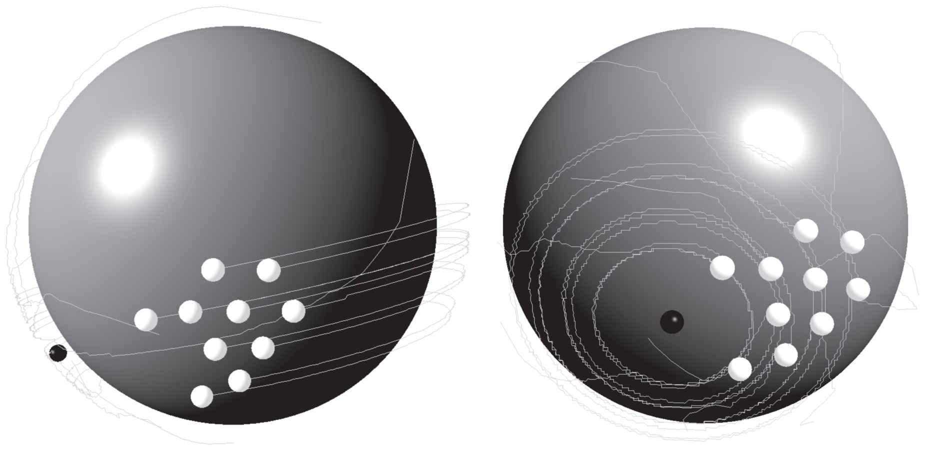


Figure 5.12: Ten-vehicle formation motion about the z axis (left), and about the x_{avg} axis (right).

5.6.3 Collision Free Reconfiguration with Avoidance of No-fly Zones

In this experiment, three aircraft are required to fly from their initial positions to given final positions. The initial positions are: $x_0^1 = [0.8659\ 0\ -0.4999]^T$, $x_0^2 = [0.4165\ -0.5721\ 0.7071]^T$, $x_0^3 = [-0.5878\ -0.809\ 0]^T$. The desired final positions are: $x_f^1 = [-0.433\ -0.7499\ 0.4999]^T$, $x_f^2 = [-0.2939\ -0.9045\ -0.309]^T$, $x_f^3 = [0.9393\ -0.3052\ 0.1564]^T$. For intervehicle collision avoidance, they are expected to maintain a minimum safety distance of $r = \cos 10°$ units. Five no-fly zones are imposed on the aircraft at the following positions: $x_{obs}^1 = [0.5237\ -0.7208\ 0.454]^T$, $x_{obs}^2 = [0.2939\ -0.9045\ -0.309]^T$, $x_{obs}^3 = [0\ -0.9877\ 0.1564]^T$, $x_{obs}^4 = [0.5878\ -0.809\ 0]^T$, $x_{obs}^5 = [0\ -0.9511\ 0.309]^T$. The radii of the no-fly zones are equal to r. Therefore $\beta^{ij} = \alpha^{ij} = 10° \ \forall\, i, j\ (i \neq j)$ for this experiment. The result is shown in Figure 5.13.

The strategy demonstrated in this simulation can be practically applied to the problem of *crossing traffic* of multiple aircraft, multiple satellite reconfiguration, and reconfiguration of multiple watercraft.

5.7 Concluding Remarks

Some technical issues about this approach are considered in this section. First, because of numerical problems the consensus position achieved during rendezvous using this approach does not converge to the exact average consensus value. This is due to the normalization process using the norm constraint, and because the norm constraint is required for this method of collision avoidance to work, it cannot be avoided.

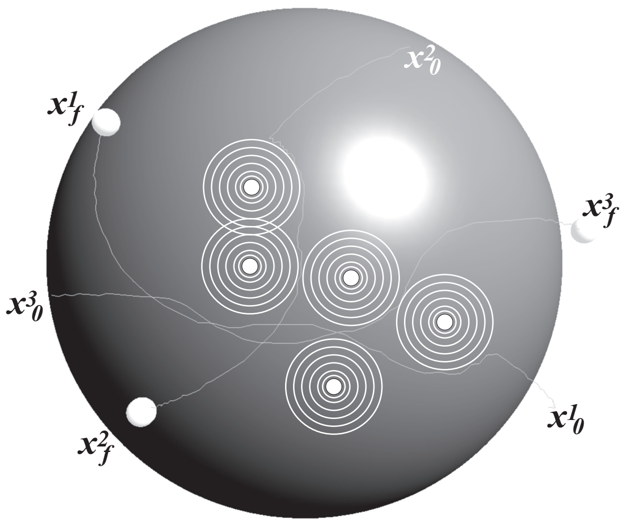


Figure 5.13: Three-vehicle reconfiguration with collision avoidance and avoidance of no-fly zones.

Secondly, for the Q-CAC algorithm, it will be desirable to be able to define upper bounds for the constraints, just as we can define lower bounds, i.e. for any $\theta^{ij} \geq \beta^{ik}$, one should be able to define $\theta^{ij} \leq \eta^{il}$, for some $\eta^{il} > \beta^{ik}$.

Thirdly, at this stage the only formation that converges to a formation with constant motion is the one achieved using the cyclic graph. For the complete graph topology, the formation comes to a halt at steady state, only then can the rigid body be controlled on the sphere, and this method does not seem to be sufficiently elegant. One may consider using double integrator dynamics for the objective function (5.4.1) which converges to a constant velocity. Also, one may combine (5.4.1) and (5.6.1) to form a new objective function that converges to a constant velocity formation.

Lastly, it will be desirable to include the concept of relative offset quaternions of Chapter 4 to this framework. This will enable one to easily define any desired formations without using computationally costly avoidance constraints every time step. However, the difficulty in doing this is that computations on quaternions are different from that on Cartesian vectors, therefore obtaining the exact desired results will not be easy. For example, the centroid of formation achieved using unit quaternions is always different from that obtained using their equivalent unit vectors, except in the first octant.

The issues considered in this section will be investigated as a part of future work.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The focus of this research was on the development of new navigation and control algorithms for multi-agent systems, such as multiple autonomous vehicles and mobile robots systems. The specific work was on the development of new efficient algorithms, for multi-path planning and multi-rigid body constrained attitude control, by combining concepts of consensus and optimization theories. The combination of concepts of consensus and optimization theories in Chapters 3, 4 and 5 provide the control models for representing the constrained control problem stated in Problem 1.1.

The main aim of Problem 1.2 was to solve some of the complexity problems associated with some of the current approaches to multi-path planning and multiple rigid body attitude control. Another aim was to solve the mathematical problems that have previously made it impossible to apply consensus theory directly with quaternions for attitude control of multiple rigid bodies.

The problems inherent in the main approaches were studied, with the intent of finding what factors limit the performance of the systems, especially complexity, and how it can be reduced. A new approach was adopted which combines the simplicity and efficiency of the graph theoretic consensus protocol with an optimization approach which is based on semidefinite programming, using linear matrix inequalities. In this approach, rather than solving the multi-path planning problems by applying computationally expensive optimization processes all the way, we have adopted a two-layer approach in which a consensus process runs on top of an optimization process, which is suppressed until it is required. Applying consensus theory essentially leads to decentralization. The following advantages of the new approach were demonstrated in the experiments: (i) an increase in the number of vehicles which could be controlled in real-time when compared to some other optimization based approaches; (ii) an increase in the number of rigid bodies (such as SC) whose attitudes could be controlled in real-time under constraints, when compared to similar optimization based approaches; (iii) a drastic reduction in convergence time for multi-path planning and multi-rigid body attitude control when compared to other similar optimization based approaches.

To this end, this research has made the following new contributions to the field of multi-path planning and multi-body constrained attitude control:

- The introduction of a new LMI-based approach to collision avoidance in 2D and 3D spaces.
- The introduction of a consensus theory of quaternions by applying quaternions directly with the consensus protocol for the first time.
- A consensus and optimization based path planning algorithm for multiple autonomous vehicle systems navigating in 2D and 3D spaces.
- A proof of the consensus protocol as a dynamic system with a stochastic plant matrix.
- A consensus and optimization based algorithm for constrained attitude synchronization of multiple rigid bodies.
- A consensus and optimization based algorithm for collective motion on a sphere.

Satisfactory solutions to Problems 1.1 and 1.2 were demonstrated in the simulation results presented in the thesis.

As a final note on the conclusions, this research has touched the three main navigation aspects of control of multiple vehicle systems, which are: control in 2D and 3D spaces; control on a sphere; attitude control. However, the consensus based approaches developed in this work are not limited to multi-path planning for multiple vehicles. They can also be applied to other areas of multi-agent control systems, where decisions and actions taken are based on agreement (or consensus) among communicating agents.

6.2 Future Work

Among the plethora of multi-path planning algorithms available in the literature, a new algorithm that can navigate a larger number of vehicles in a smaller space, at a faster speed, and without collisions, will still always be desired. Simply put, the navigation problem has not been completely solved, and it remains a challenge. Therefore it will be worthwhile to compare the consensus and Q-CAC based framework developed in Chapter 3 with some of the best existing approaches, based on these benchmarking criteria. In fact, comparing all of the existing approaches based on these benchmarks is a significant research effort worth undertaking. This will also provide information for researchers to know what has been done in navigation research, and what remains and needs to be done.

Due to the unavailability of equipment, practical hardware implementation of the consensus and Q-CAC based attitude control algorithm developed in Chapter 4 was not done in this thesis, and we hope to do it in the future. Moreover, the algorithm was developed for a simplified rigid body model. It will be desirable to extend it to complex rigid bodies with flexible rotating appendages, e.g. spacecraft with rotating rigid parts, or cooperating manipulators with multiple joints. Moreover, we have yet to determine the maximum number of rotating rigid bodies that can cooperate without incurring collisions. These will require significant research efforts in the future.

The constrained path-planning algorithm on the sphere, which was developed in Chapter 5, still remains to be further developed. At this stage, the algorithm requires both the consensus optimization and the Q-CAC optimization processes to run every time step, with the result that the speed of the algorithm is fairly slow for a relatively large number of vehicles. However, reconfiguration of up to ten vehicles on the sphere runs fairly fast. We have yet to ascertain the maximum number of vehicles that can be allowed for real-time path planning, which can also be implemented on real vehicles. Also, the numerical problem of error in consensus position resulting from the norm constraint in the LMI, remains to be solved.

For the hardware implementation in Appendix A, we have implemented simple PID controllers on UAVs. From the results presented, more work needs to be done to develop better controllers because the PID controller has not proven robustness to disturbance resulting from wind gusts that emanate from drone-drone physical interactions. It is a challenge to design a controller that can stabilize more than one interacting drone (especially light weight drones like the AR.Drone), in a *constrained flying space* and in the presence of wind gusts emanating from all of them. In the future we hope to study the effects of wind gusts and design new mixed sensitivity controllers, or multivariable controllers that take this interaction into account.

Finally, networking and communication issues have not been addressed in this work and we hope to do it in the future.

Appendices

Appendix A

Hardware Implementation

In order for a vehicle to follow the trajectory generated by a path planner, controllers have to be employed. This appendix is on the development of a tracking controller for control along the planned position trajectories, for the multi-path planning algorithm developed in Chapter 3. The controller was implemented on quadrotor unmanned aerial vehicle (UAV) platforms. To enable the implementation, an overhead camera vision system was set up and connected to a control desktop computer running the RoboCup[1] small sized league vision system[2], for image processing. This was used to provide real-time measurements of the positions and attitudes of the vehicles. The state updates enable the controller to track a given trajectory. A wireless local area network was also set up for communication between the drones and the control computer which also runs the vision processing software and the controllers. Experimental results are provided for waypoints visiting, and for a minimal case of consensus based path planning with Q-CAC based collision avoidance.

[1]http://www.robocup.org/
[2]http://code.google.com/p/ssl-vision/

A.1 Introduction

The control problem is that of driving the state of a dynamic system from one state $x(t_k)$ at time t_k, to another state $x(t_{k+1})$ at time t_{k+1}, accurately, and in finite time. For our implementation, the system is a quadrotor unmanned aerial vehicle (UAV). A quadrotor helicopter or quadcopter is an aerial vehicle propelled by four rotors. The vertical take-off and landing (VTOL) capability of the quadrotor and its small size make it a suitable choice to use for such tasks as: traffic monitoring; crime monitoring; search and rescue; security and surveillance (Faigl *et al.*, 2010); cooperative manipulation and transportation (Michael *et al.*, 2011); human-machine interaction (Ng and Sharlin, 2011); sports assistance (Higuchi *et al.*, 2011); and military reconnaissance. Currently, the quadcopters can quickly perform complex manoeuvres (Mellinger *et al.*, 2010), and navigate autonomously in structured and unstructured environments (Achtelik *et al.*, 2009; Bills *et al.*, 2011; Blosch *et al.*, 2010).

However, the quadrotor dynamics is inherently unstable and difficult to control. In addition to this problem, the AR.Drone[3] quadrotor which was used for this work is extremely light weight (420g with the outdoor hull), compared to its size (660mm in diameter). It has been previously identified that the light weight of the AR.Drone does not make it suitable for use in medium or high disturbance environments (Schmidt, 2011), and this has been confirmed in this work. Moreover, when any extra load is attached to the drone to enhance the weight, it quickly becomes unstable.

Since the advent of the AR.Drone, an affordable quadrotor UAV with an array of sensors, there has been some efforts in the research community to make it a suitable research platform for robotics and control systems. This work is one of the first attempts to develop the AR.Drone to be used for networked multi-agent control experiments.

Next, we consider examples of some recent controller designs for quadrotor UAVs. Bouabdallah *et al.* (2004) performed a comparison of classical control using PD and PID controllers, and adaptive optimal control using LQR controllers, on a quadrotor UAV. Their results showed that the performance of the classical PID controller was better than the modern optimal controller using LQR. They also showed that the PID controller was not robust

[3]http://www.parrot.com

in the presence of disturbance. Other successful PID controller designs for quadrotor UAVs have been reported (Gurdan *et al.*, 2007; Hoffmann *et al.*, 2008; Krajnik *et al.*, 2011). Alexis *et al.* (2010*b*) developed a model predictive control (MPC) based constrained finite time optimal controller (CFTOC), for attitude setpoint manoeuvre of a quadrotor operating under severe wind conditions. The CFTOC was based on the multiparametric toolbox of Kvasnica *et al.* (2004). Measurement of the quadrotor states was provided by the utilization of an Xsens MTi-G[4] attitude and heading reference system (AHRS). The induced wind gust velocities were measured using a rotary vane anemometer. The resulting controller has been validated in experimental studies with a prototype DraganFlyer[5] quadrotor, modelled as a set of switching linear piecewise affine systems (PWAs). A similar approach was presented in Alexis *et al.* (2010*a*) for trajectory tracking of a quadrotor, under wind gusts, however only simulation results were presented.

Successful machine learning and artificial intelligence (AI) control schemes have been reported (Bills *et al.*, 2011; Ng and Sharlin, 2011). However, none of the aforementioned works was done for multiple drones in a tightly spaced environment.

It has been proven that a classical PID controller is sufficient for controlling the quadrotor UAVs in the absence of high disturbance. Indeed, we developed and implemented two different controllers to test hover and tracking on the AR.Drones, one using an MPC controller, and another using a PID controller. Our results show that the PID controller had a better performance, with a faster response time, better tracking and hover. Therefore, we have adopted a classical PID control technique for this work.

The rest of the appendix is organized as follows. In Section A.2 a brief background to the work is presented, and in Section A.3 the control problem is stated. PID controller synthesis for the AR.Drone quadrotor is presented in Section A.4. Experiments overview together with the hardware and software setup are in Section A.5. The experiment results are presented in Section A.6 and concluding remarks follow in Section A.7.

[4]http://www.xsens.com/en/general/mti-g

[5]www.draganfly.com

A.2 Background

The dynamics of a vehicle navigating in $\mathbb{R}^3$ can be approximated by the motion of a simple point mass, where the state variables are the position and velocity $[x(t), y(t), z(t), \dot{x}(t), \dot{y}(t), \dot{z}(t)]^T$, and the control inputs are acceleration $[a_x, a_y, a_z]^T$. Using this model, the continuous time dynamic system is given as

$$\underbrace{\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \\ \ddot{x}(t) \\ \ddot{y}(t) \\ \ddot{z}(t) \end{bmatrix}}_{\dot{\mathbf{x}}(t)} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix}}_{\mathbf{x}(t)} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} a_x(t) \\ a_y(t) \\ a_z(t) \end{bmatrix}}_{\mathbf{u}(t)} \tag{A.2.1}$$

$$\mathbf{y} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{C}} \mathbf{x}.$$

It has been shown that the dynamics (A.2.1), subject to two-norm constraints, is a good approximate model for vehicle motion, including limited turn-rate vehicles such as aircraft (Kuwata, 2003; Richards, 2005). The zero-order hold (ZOH) discrete time system equivalent of (A.2.1) is given as

$$\underbrace{\begin{bmatrix} x(k+1) \\ y(k+1) \\ z(k+1) \\ \dot{x}(k+1) \\ \dot{y}(k+1) \\ \dot{z}(k+1) \end{bmatrix}}_{\mathbf{x}(k+1)} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x(k) \\ y(k) \\ z(k) \\ \dot{x}(k) \\ \dot{y}(k) \\ \dot{z}(k) \end{bmatrix}}_{\mathbf{x}(k)} + \underbrace{\begin{bmatrix} \frac{(\Delta t)^2}{2} & 0 & 0 \\ 0 & \frac{(\Delta t)^2}{2} & 0 \\ 0 & 0 & \frac{(\Delta t)^2}{2} \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} a_x(k) \\ a_y(k) \\ a_z(k) \end{bmatrix}}_{\mathbf{u}(k)} \tag{A.2.2}$$

$$\mathbf{y}(k) = \underbrace{\mathbf{C}}_{[\mathbf{I}_3\ \mathbf{0}_3]} \mathbf{x}(k),$$

where $\mathbf{I}_3$ is a 3×3 identity matrix, $\mathbf{0}_3$ is a 3×3 zero matrix, k is the discrete time step, and Δt is the time interval. Based on this model, one can design any choice of controllers to stabilize the system, and simulate with different parameters to fine tune the system.

A.3 Problem Statement

Given the current state of a vehicle $\mathbf{x}(t)$, and the current desired reference trajectory or set point vector $\mathbf{x}_d(t) = [x_d(t)\ y_d(t)\ z_d(t)\ \dot{x}_d(t)\ \dot{y}_d(t)\ \dot{z}_d(t)]^T$, the aim of the controller is to drive the system (A.2.1) or (A.2.2) from $\mathbf{x}(t)$ to converge *smoothly* to $\mathbf{x}_d(t)$, in the presence of noise and disturbance. The difference between the current state and set point, or desired trajectory, is defined as

$$\mathbf{x}_e(t) = \mathbf{x}_d(t) - \mathbf{x}(t),$$

and therefore, the error dynamics is

$$\dot{\mathbf{x}}_e(t) = \mathbf{A}\mathbf{x}_e(t) + \mathbf{B}(\mathbf{u}(t) - \mathbf{u}_d(t)). \tag{A.3.1}$$

It is assumed that the state $\mathbf{x}(t)$ can be measured accurately and $\mathbf{x}_d(t)$ is always known for all t. The control vector $\mathbf{u}_d(t)$ is unknown but treated as disturbance. Based on the error dynamics (A.3.1), the initial problem of driving $\mathbf{x}(t)$ to $\mathbf{x}_d(t)$ is transformed to an equivalent problem of driving the states of (A.3.1) to zero.

The imperfection of the physical system, and other environmental factors such as disturbance and noise, makes the control problem difficult. For our multi-agent control problem, we are faced with the extra problem of stabilizing the light weight AR.Drones, in the presence of wind gusts emanating from each of them, in the small operational space $(3m \times 4m)$ used for the experiments. In our experiments, we found it particularly difficult to get more than one drone to maintain stable hover, when they are spaced apart by less than $3m$. When we tried to perform hover for two drones, both of them veered off in opposite directions. With these problems, the need to design high level stabilizing feedback controllers becomes obvious.

A.4 PID Controller Synthesis for Position Control

In this section, we first consider the principles of quadrotor motion, then a PID controller design for the AR.Drone is presented.

A.4.1 Quadrotor Motion

Quadrotor motion is usually obtained by rotating each pair of opposite motors in the same direction, and the two pairs of motors are rotated in opposite directions. For the AR.Drone, the front right and rear left motors, are rotated counterclockwise, and the front left and rear right motors are rotated clockwise, as shown in Figure A.1.

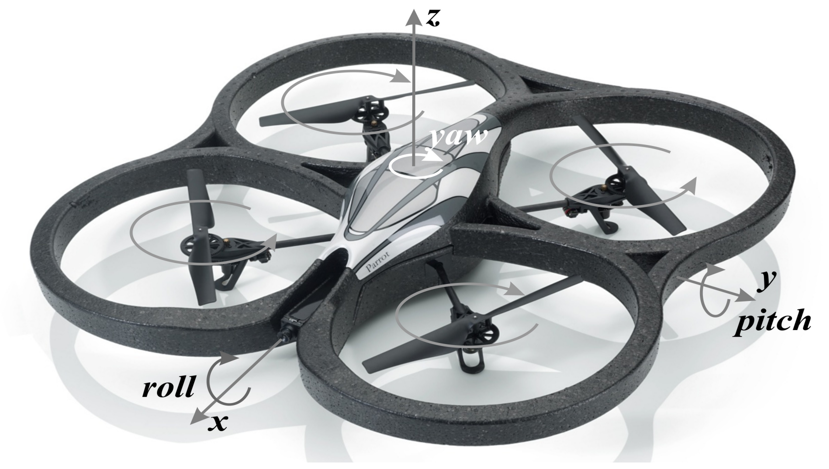


Figure A.1: Quadrotor motion.

To obtain upward motion, the four motors are rotated with an equal magnitude of velocity. Forward motion is obtained by making the magnitude of the velocities of the two front motors equal but less than the magnitude of the velocities of the two rear motors, which are also made equal. This also corresponds to forward *pitch*, and this process is reversed to obtain backward motion, or negative pitch. To move or *roll* left, the magnitude of the velocities of the two left motors are made equal but less than the magnitude of the velocities of the two right motors, which are also made equal. This is reversed to pitch right, or move right. To rotate on the spot, or yaw counterclockwise, the velocity magnitude of the front right and rear left motors are made equal but larger than the velocity magnitude of the front left and rear right motors, which are also made equal. This process is reversed to obtain clockwise yaw.

For the AR.Drone, there are already low level motor controllers, and direct motor control has been abstracted and excluded from users. The drones can be controlled by sending four basic commands: pitch, which also corresponds to forward or backward motion; roll, which also corresponds to left or right motion; altitude; and yaw. All four controls are independent, and so, four independent PID controllers can be designed for each of the controls.

Since a quadrotor is an omnidirectional vehicle, one can control its motion in any direction without having to control the yaw rotation. For our experiments, there was no explicit need to control the yaw angle. Due to the limitations of our single overhead camera measurement system, we decided to control all of the drones at the same altitude. We allowed the low level altitude controller supplied with the drone to control the altitude. A PID controller was designed, which can be used for independent control of the x and y positions of the drone. Since the low level altitude controller is fairly stable, we assumed that controlling the x and y positions is sufficient to maintain stable hover and trajectory following.

A.4.2 PID Controller Synthesis for Quadrotor Motion Control

Consider the double integrator dynamic model of (A.2.1). The equivalent transfer function of the state space system is

$$G(s) = \frac{1}{s^2}. \tag{A.4.1}$$

Testing the response of this system to a unit step input, one can see the requirement for a controller. A classical proportional-integral-derivative (PID) controller consists of the sum of three functions of the error $\mathbf{x}_e(t)$ at time t: the proportional error $e_p(t)$; the integrated error e_i; and the derivative error e_d. Therefore, the original system will be augmented with an integrator $\frac{1}{s}$, and a derivative s. The PID transfer function is defined as

$$G_{PID}(s) = K_p + K_i\frac{1}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}, \tag{A.4.2}$$

where the proportional gain K_p, the integrated gain K_i, and the derivative gain K_d, are constant gains which are selected to fine tune the system. Ignoring disturbance and noise factors, the closed loop system transfer function is

$$G_{CL}(s) = \frac{G(s)G_{PID}(s)}{1 + G(s)G_{PID}(s)} = \frac{K_d s^2 + K_p s + K_i}{s^3 + K_d s^2 + K_p s + K_i}. \tag{A.4.3}$$

However, in our dynamic multi-agent systems scenario, external disturbance is a critical issue that should be considered. In addition, the measurement produced by the camera system is sometimes susceptible to corruption due to measurement noise. Sometimes the measurement can be misleading, and sometimes there are no measurements at all. The PID control system that also includes disturbance and noise factors is illustrated in Figure A.2.

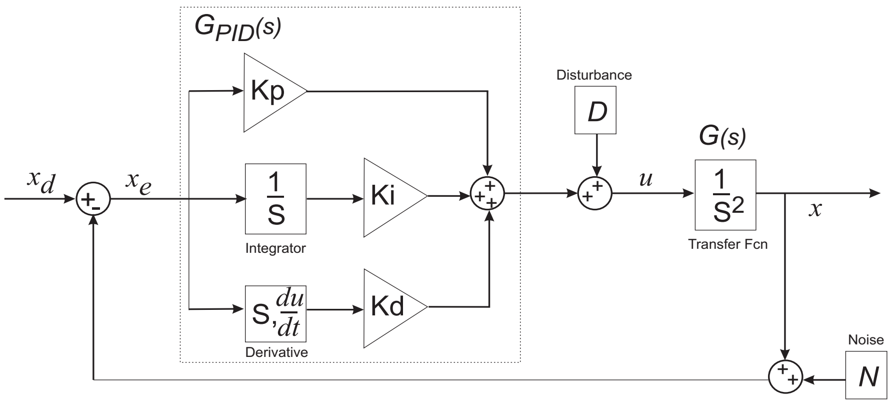


Figure A.2: PID control system for (A.2.1).

To implement the PID control practically, the control is computed as the sum of three errors $e_p(t)$, $e_i(t)$, and $e_d(t)$, which are functions of x_e, every time step, as follows:

$$\begin{aligned}
e_p(t) =& K_p x_e(t),\\
e_i(t) =& K_i(\underbrace{x_e(0) + x_e(1) + \cdots + x_e(t)}_{\int x_e dt}),\\
e_d(t) =& K_d(\underbrace{x_e(t) - x_e(t-1)}_{\frac{dx_e}{dt}}),\\
u(t) =& e_p(t) + e_i(t) + e_d(t).
\end{aligned}$$

Assuming $D = 0$ and $N = 0$, the unit step response of the system in Figure A.2, with $K_p = 0.9$, $K_i = 0.2$, and $K_d = 1$, is shown in Figure A.3.

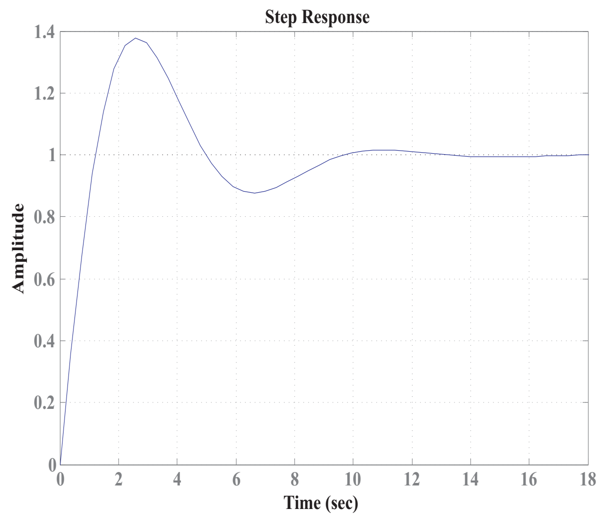


Figure A.3: Step response of (A.4.3) with $K_p = 0.9$ and $K_i = 0.2$, $K_d = 1$.

A.5 Experiments Overview and Systems Setup

In this section, an overview of the experiments, together with the hardware and software setup, are described.

A.5.1 Experiments Overview

The purpose of the experiments performed was to demonstrate real-time path planning and following, with collision avoidance. The consensus-based multi-path planning with Q-CAC collision avoidance algorithm, developed in Chapter 3, was used for the path planning, while each drone runs the PID controller (above) to independently control x and y positions, in order to follow the planned paths. The path planning, collision avoidance, and path following, are run simultaneously.

A.5.2 Hardware and Software Setup

The setup used in the experiments consists of the following: (i) three AR.Drones, from Parrot[6]; (ii) a desktop PC running Ubuntu Linux, Matlab, Java, ssl-vision, gcc and g++, and the drones control software which was written in Java; (iii) a Firewire AVT[7] Stingray camera connected to the PCI port of the PC; (iv) a Sitecom[8] 802.11g wireless router.

A.5.2.1 AR.Drone Platform

The AR.Drone, introduced in Section A.4.1, is an electrically powered quadcopter manufactured by Parrot, and intended for augmented reality games. The entrance of the AR.Drones into the market since March 2010 provided an affordable platform for students and research groups to conduct multi-agent systems research. This is because of the affordable price of the drones, when compared with other high end platforms such as the DraganFlyer[9]. Comparatively, the AR.Drone provides enough sensors to conduct research in autonomous navigation studies. And when the research does not involve the hardware development, students and researchers can concentrate on the necessary aspects and rely on the platform for rapid development, testing, and deployment of control algorithms.

The drone is built of a carbon-fiber support structure and a plastic body. Each drone ships with a safety indoor and outdoor removable hull (see Figure A.1). The actuators are four high-efficiency brushless motors. The sensors suit consists of the following: two cameras, one in front and facing forward, and the other one below the drone and facing downwards; two sonars below the drone used together as an altimeter for the drone; an IDG-400 2-axis gyroscope and 3-axis accelerometer, which are used to provide pitch and roll data; and a XB-3500CV high precision gyroscope, used to measure yaw. These sensors provide a 6-degree-of-freedom inertial measurement unit (IMU) for the drone. One LIPO battery is also supplied with each drone, and each battery can provide enough energy for up to 13 minutes of continuous flight.

[6]www.parrot.com
[7]http://www.alliedvisiontec.com/
[8]http://www.sitecom.com/
[9]http://www.draganfly.com

The AR.Drone control computer is based on the ARM9 processor running at 468MHz, with 128 MB of DDR RAM running at 200MHz. Each drone comes with an 802.11g wireless fidelity (WiFi) device and an installed software interface, which allows it to communicate with another WiFi enabled device, such as a WiFi enabled computer, via an ad hoc WiFi network. The API allows one to set required states of the drone, and also provides access to preprocessed sensory measurements and images from the onboard cameras.

However, this ad hoc WiFi setup makes it difficult to connect more than one AR.Drone in a wireless managed network infrastructure, because the manufacturers have restricted the access to the drones hardware via only the WiFi connection and any other method of access voids the warranty. Because of this limitation, the AR.Drones have not been used previously in a networked multi-agent team of more than one AR.Drone.

The control board of the AR-Drone runs the BusyBox based GNU/Linux distribution with the 2.6.27 kernel. The internal software of the drone provides communication and some assisted manoeuvres such as take-off and landing. After being switched on, an ad hoc WiFi network appears, and an external computer might connect to it using a fetched IP address from the drone dynamic host configuration protocol (DHCP) server. Using this IP address, one can telnet into the drone and run a script file that enables the drone to disconnect from the current ad hoc network, and connect to a managed infrastructure, e.g. to the Sitecom router. The script file can be created and downloaded once and for all, into all of the drones, via ftp. With several drones connected to a wireless router running a DHCP server, one has a network of AR.Drones. The DHCP server assigns new IP addresses for the drones. With this arrangement, it is safe to network the drones without voiding the manufacturer's warranty. However, one has to manually connect to all of the drones and run the script file every time the drones startup, and this process can be tedious and time consuming.

Once the multi-agent network is setup, a computer in the network can communicate with each of the drones, using the interface provided by the manufacturer on each drone. The interface communicates via three channels, each with a different user datagram protocol (UDP) port. Drone configuration and actuator commands are sent over the *command* channel via UDP port 5556. Navigation data, such as yaw, pitch, roll, altitude, battery state, and 3D speed estimates, are received over the *navdata* channel, via UDP port 5554. Video data from the cameras are received over the *stream* channel, via UDP port 5555. The frequency of each of the channels is 30Hz.

With this simple software setup, any programming language that supports network programming (such as C, C++, Java, and Visual Basic) can be used to develop control and interaction software for the AR.Drones. However, many potential users are inundated by the volume of the C based API programs that are supplied with the drones. Users find it extremely difficult to start working with the drones, because understanding the API requires looping to and from so many C program files. We consider this unnecessary, because the network-channel framework should make things simpler. Indeed based on a user supplied single Java class of a few hundred lines of code for keyboard control of the AR.Drone, we have developed just a few classes of Java programs to control the AR.Drones in a network. We hope that these developments will assist in using the AR.Drones as a suitable cost effective platform for conducting research in multi-agent systems.

A.5.2.2 Ssl-vision System

To obtain external measurements of the positions and yaw angles of the drones, an AVT Stingray camera is connected to the control computer via the Firewire 800 PCI interface, which can provide visual data frames at up to 60 frames per second at a resolution of 780x580 pixels, to the computer for processing. Image processing is done with the RoboCup ssl-vision software. After processing, ssl-vision returns position and yaw data of all the vehicles detected. The returned data is encoded using Google protocol buffers[10] and multicasted via a UDP multicast server written in C++.

For the vision system to detect individual vehicles each of the vehicles must carry an appropriate ssl team marker image, which is a set of color patterns placed on top of the vehicles that uniquely identifies them (see Figure A.4).

The ssl-vision system was designed for soccer playing robots navigating on the plane

[10]http://code.google.com/apis/protocolbuffers/

and the maximum height required for the robots in order for the vision system to provide accurate position measurements is $140mm$. This setup was not developed for tracking of flying vehicles, and the drones cannot be flown at $140mm$. Since the drones do not have global positioning system (GPS) capability to use for outdoor tracking, the ssl-vision system was used because it was the only available facility we had at the time of the experiments to provide position measurements. A suitable indoor 3D motion tracking system such as Vicon motion capture system[11] would be preferred for use in future.

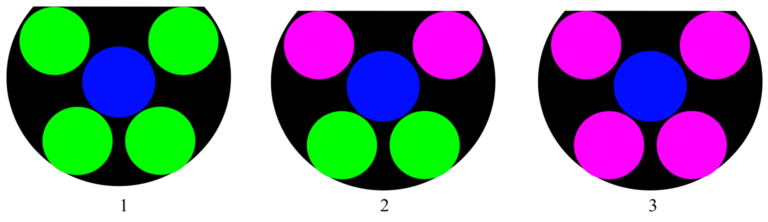


Figure A.4: Vehicle identification markers.

To make the ssl-vision useful for our experiments, we decided to fly the drones at a low and constant altitude of $500mm$. Because of this height difference and height variations that occurred due to perturbations on the altitude of the drones while they were flying, there were small errors in the position measurements. This setup also limited the kinds of manoeuvres that we could test in 3D. However, the setup was fairly adequate to perform a few useful experiments.

A.5.2.3 Control Station and Software

The control PC serves as a centralized control server, for feedback control of the motion of the drones. Centralized control was adopted in the implementation because of the limitations of the drone hardware. At this stage, user programs cannot be run on the AR.Drone hardware, one can only send commands and receive sensor data via WiFi connectivity. The wireless local area network (WLAN) was built on a Sitecom 150N 802.11g wireless router sampling at 150Mbps. The PC runs the consensus based multi-path planner with the Q-CAC avoidance algorithm which has been implemented in Matlab. The PC also runs a Java

[11]http://www.vicon.com/

control program. The control program includes: separate PID feedback controllers for each drone, implemented in Java; a set of Java classes used to send control data to the drones, and obtain real time position and yaw feedback information from ssl-vision, via a UDP multicast client class; a Java function that calls the Matlab based function for consensus with the Q-CAC avoidance trajectories. Vision data obtained from ssl-vision are decoded using Google protocol buffers. Other information such as velocity $\dot{\mathbf{x}}$, altitude z, and battery state, are obtained from the drones directly into Java control program. Communication between Java and Matlab was established via matlabcontrol[12]. The hardware and software setup is illustrated in Figure A.5.

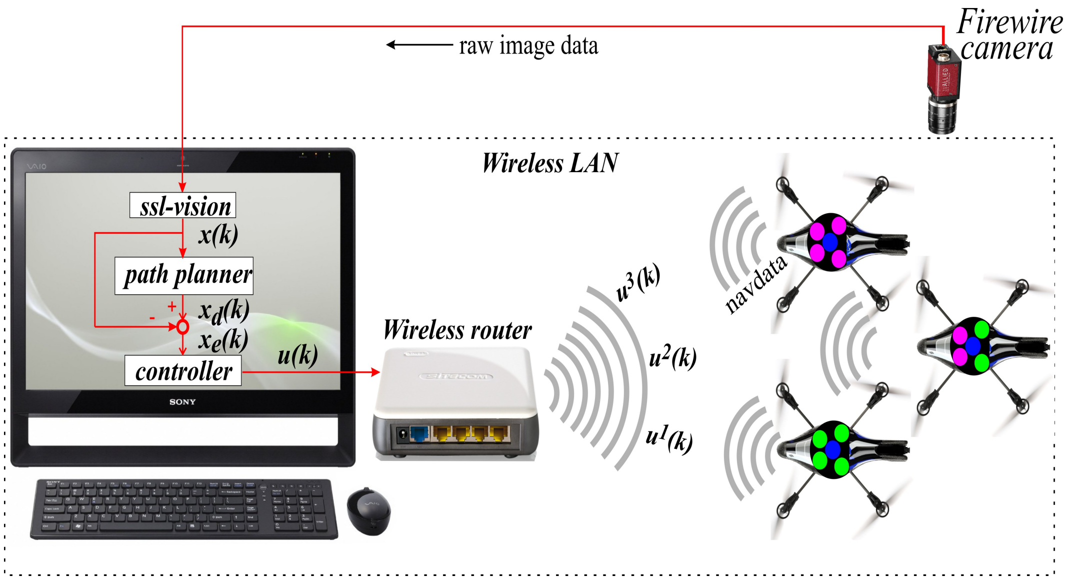


Figure A.5: Hardware and software setup.

The real-time path planning and tracking control procedure is implemented as follows:

- Step 1: initial positions $\mathbf{x}(0)$ and desired final positions $\mathbf{x}(f)$, are given to the multi-path planner algorithm. If $\|\mathbf{x}(f) - \mathbf{x}(0)\| > \epsilon$, where $\epsilon > 0$ is a small error tolerance margin, then proceed to the next step, otherwise the goal has been reached.

- Step 2: the multi-path planner algorithm generates the next collision free set of trajectories and passes these as set points $\mathbf{x}_d$ to the controllers.

[12]http://code.google.com/p/matlabcontrol/

- Step 3: current position $\mathbf{x}(k)$ and yaw data $\theta(k)$ are obtained from ssl-vision. $\mathbf{x}(k)$ are passed to the controllers, and $\theta(k)$ are passed to $\mathbf{R}$, a rotation matrix that transforms the control inputs to the drone coordinates.
- Step 4: while $\|\mathbf{x}_e(k)\| > \epsilon$ for some $\epsilon > 0$, the controllers generates a set of control inputs $\mathbf{u}(k)$, sends them to the drones, and k is incremented by one.
- Step 5: when $\mathbf{x}_e(k) \approx \epsilon$, the current $\mathbf{x}(k)$ is the new set of initial positions, and the algorithm goes back to step 1.

A simplified model of the above process is shown in Figure A.6. Since this is not a multivariable control system, the control algorithm runs two separate PID controllers for each state x and y, for each drone to be controlled.

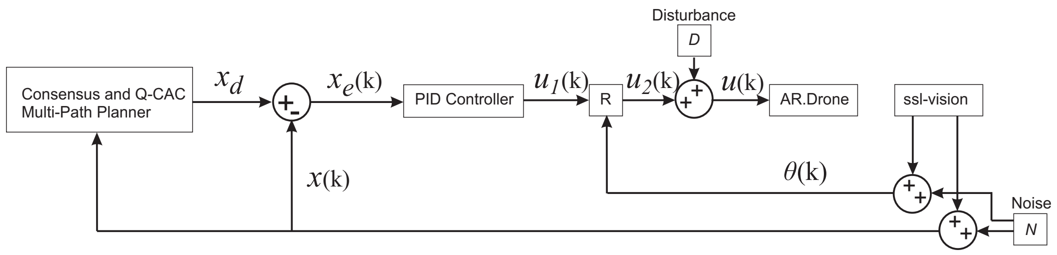


Figure A.6: Multi-path planning and tracking control system.

A.6 Experiment Results

Because a single camera was used for supervisory data acquisition, the flight space of the drones was limited to the field of view of the camera. Using the ssl RoboCup soccer pitch dimensions, the drones were constrained to fly within the space of half of the pitch, which has the dimension $3025mm \times 4050mm$. The diameter of each drone was $660mm$. The limited space for flight imposed serious constraints on the experiments, thereby limiting the number of drones that could be controlled simultaneously, and the kinds of experiments that could be performed successfully.

Before we could use the drones to test the multi-path planning algorithms, it was desired to test the performance for way-point visiting and trajectory following. These tests could not be performed using only the built-in low level controller supplied with the drones. In fact, it was observed that if two drones approached each other for any distance less than $3m$, both drones veered off away in opposite directions. This effect was because of disturbance resulting from the wind gusts emanating from the drones, coupled with the very light weight of the drones. The PID controller enabled three drones to hover on the half pitch space when spaced apart. However, because of the high noise and light weight of the drones, we were unable to perform any collision avoidance experiments on the same space with two or more drones flying together at a time.

In this work, we did not study the effects of wind gusts, and as a result we do not yet have a model or transfer function representation for the gusts. To counter the effect of disturbance, we tried setting an upper bound on the control input that will also allow the drones to be controlled safely within the small flying space. The experiment results are presented next.

A.6.1 Waypoints Visiting

In this experiment, a single drone starts from a point $\mathbf{x}(0) = [-0.44362\ 0.00027]^T$, to visit four waypoints $A = [-0.44362\ 1.57303]^T$, $B = [-2.57429\ 1.57303]^T$, $C = [-2.61834\ -1.63512]^T$, $D = [-0.44362\ -1.63512]^T$, and then attempts to return back to $\mathbf{x}(0)$. The PID controller Figure A.2 was used for waypoints tracking, with $K_p = 0.5$, $K_i = 0.5$ and $K_d = 0.8$. The final return point was $\mathbf{x}(f) = [-0.4355\ -0.01563]^T$. The result is shown in Figure A.7.

A.6.2 Two-Vehicle Consensus with Q-CAC Collision Avoidance

In this experiment, two drones are connected in a leader-follower graph topology, where vehicle 2 is the leader. Drone 1 starts from $\mathbf{x}^1(0) = [-2.61834\ 1.63512]^T$, and attempts to find a collision free trajectory to drone 2 at $\mathbf{x}^2(0) = [-0.4436\ -1.64078]^T$. Drone 3 is on the line between drones 1 and 2 at $\mathbf{x}^3(0) = [-1.5\ 0]$ and has to be avoided. The final position of drone 1 is $\mathbf{x}_f^1 = [-0.36136\ -1.76748]$. The result for this experiment is shown in Figure A.8. In the figure the blue line is the planned trajectory, and the black line is the actual trajectory.

A.7 Concluding Remarks

The experiments presented above were successful because only one drone was allowed to fly within the available space. From the experience gained in doing this work, it is clear that the implementation of control systems for the path planning algorithms on physical platforms is not a trivial task. The knowledge gained from conducting these experiments leads to the following findings:

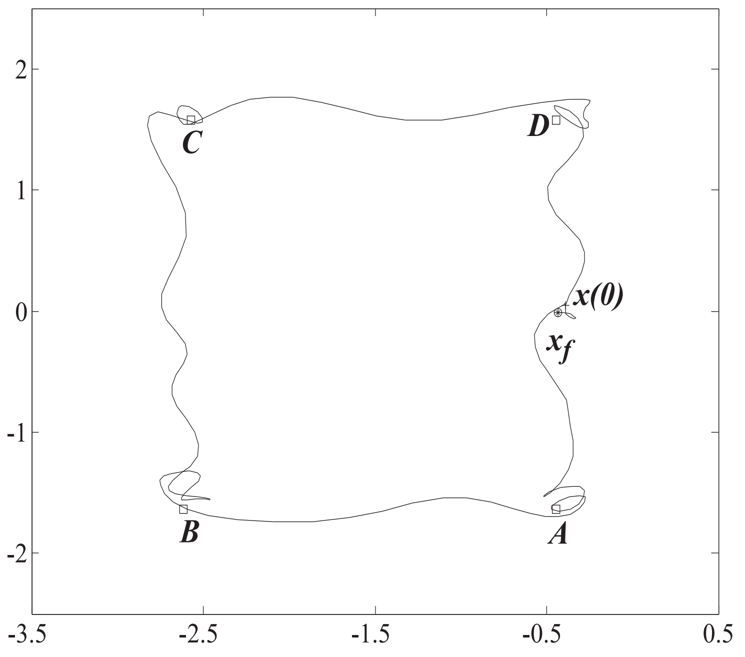


Figure A.7: Waypoints visiting.

- Because of the small flight space coupled with the extremely light weight of the drones, the maximum control input to the drone for motion in the x or y axis was set at 0.1 (1m/s). Any speed greater than this resulted in instability, such as drones overshooting the boundary of the camera perception region. However, this control effort was not sufficient to counter the force produced by the wind gusts when two drones came close to each other. The effect was that one of the drones always gets pushed off track away from the limited field of view of the camera, while the other arrives at its destination. Because of this, the safety region of the drones should be made sufficiently large, to minimize the effect of gusts because of the light weight of the drones. We believe that conducting the experiments in a bigger space should produce better results.

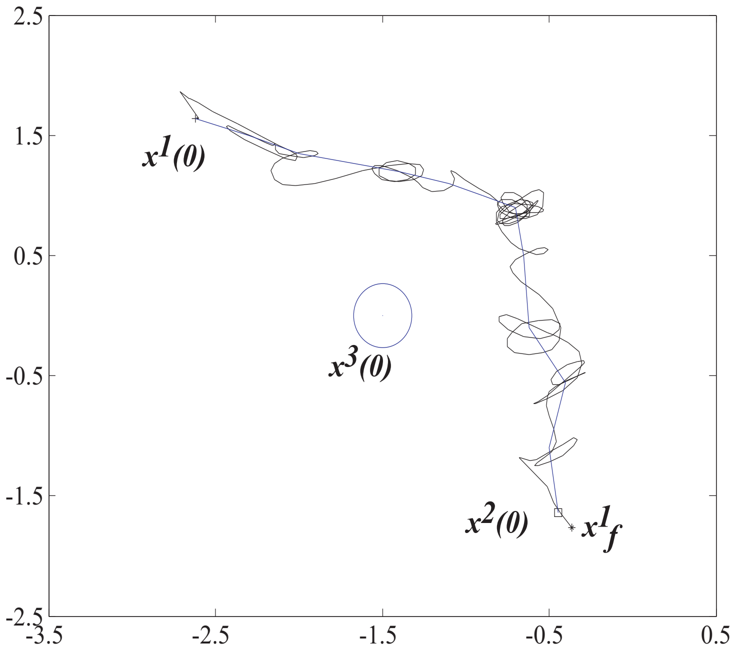


Figure A.8: Consensus with collision avoidance.

- The overhead vision system was not the ideal choice for tracking of the drones. In using the system to conduct our experiments, we observed that changes in height and attitudes of the drones resulted in packet losses, which had adverse effects on the experiments. Moreover, the lighting in the laboratory was not very well controlled, as the effects of changes in ambient light emanating from some open windows, were immediately noticed in the behavior of the drones. Most recent indoor UAV navigation experiments use reliable and proven 3D motion capture systems such as Vicon.

- Fitting the AR.Drones with GPS based AHS such as Xsens MTi-G may help to use the drones in outdoor experiments, where enough space is available. In this case, one should also consider the following: the light weight of the drones when exposed to windy outdoor environments; and how much load the drones can carry and remain stable. However, we observed that the wind gusts emanating from the drones were worse than normal outdoor conditions. Therefore, we hope to conduct future experiments in outdoor environments.

- We consider the problem of controlling the drones in a small space as a technical challenge that demands more research. We hope to conduct an in-depth study of the effects of the wind gusts, and develop gust models which can be used for a mixed sensitivity controller design. This may also help for control in outdoor environments.

List of References

Achtelik, M., Bachrach, A., He, R., Prentice, S. and Roy, N. (2009). Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments. *SPIE Unmanned Systems Technology XI*, vol. 7332.

Ahmed, A., Alexander, J., Boussalis, D., Breckenridge, W., Macala, G., Mesbahi, M., Martin, M.S., Singh, G. and Wong, E. (1998). *Cassini Control Analysis Book*. Tech. Rep., Jet Propulsion Laboratory, CALTECH, Pasadena, CA.

Alexis, K., Nikolakopoulos, G. and Tzes, A. (2010*a*). Constrained-control of a quadrotor helicopter for trajectory tracking under wind-gust disturbances. In: *Proc. IEEE ACC*, pp. 1411–1416. Baltimore, MD.

Alexis, K., Nikolakopoulos, G. and Tzes, A. (2010*b*). Constrained optimal attitude control of a quadrotor helicopter subject to wind-gusts: Experimental studies. In: *Proc. IEEE ACC*, pp. 4451–4455. Baltimore, MD.

Altman, E., Basar, T. and Srikant, R. (1999). Congestion control as a stochastic control problem with action delays. *Automatica*, vol. 35, pp. 1937–1950.

Ando, H., Oasa, Y., Suzuki, I. and Yamashita, M. (1999). Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE J. Robot. Autom.*, vol. 15, no. 5, pp. 818–828.

Aoki, M. (1968). Control of large-scale dynamic systems by aggregation. *IEEE Trans. Autom. Control*, vol. 13, no. 3, pp. 246–253.

Balch, T. and Arkin, R.C. (1998). Behavior-based formation control for multirobot teams. *IEEE J. Robot. Autom.*, vol. 14, no. 6, pp. 926–939.

Barret, G. and Lafortune, S. (2000). Decentralized supervisory control with communicating controllers. *IEEE Trans. Autom. Control*, vol. 45, pp. 1620–1638.

Barriere, L., Flocchini, P., Fraigniaud, P. and Santoro, N. (2005). Election and rendezvous in fully anonymous networks with sense of direction. *Theory of Computing Systems*.

Bauso, D., Giarre, L. and Pesenti, R. (2003). Distributed consensus protocols for coordinating buyers. In: *Proc. 42nd IEEE Conference on Decision and Control*, vol. 1, pp. 588–592.

Beard, R., Lawton, J. and Hadaegh, F. (2001). A coordination architecture for spacecraft formation control. *IEEE Trans. Control Syst. Technol.*, vol. 9, no. 6, pp. 777–790.

Beard, R.W., McLain, T.W., Nelson, D.B., Kingston, D. and Johanson, D. (2006). Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs. *Proc. of the IEEE*, vol. 94, no. 7, pp. 1306–1324.

Bhattacharya, R., Fung, J., Tiwari, A. and Murray, R.M. (2004). Ellipsoidal cones and rendezvous of multiple agents. In: *Proc. IEEE CDC*, pp. 171–176. Atlantis Paradise, Bahamas.

Biggs, N. (1974). *Algebraic Graph Theory, Cambridge Tracks in Mathematics*. Cambridge University Press, Cambridge, UK.

Bills, C., Chen, J. and Saxena, A. (2011). Autonomous MAV flight in indoor environments using single image perspective cues. In: *Proc. IEEE ICRA*, pp. 5776 – 5783. Shanghai, China.

Blackwood, G., Lay, O., Deiningec, B., Gudima, M., Ahmed, A., Duren, R., Noeckerb, C. and Barden, B. (2002). *The StarLight mission : a formation-flying stellar interferometer*. Tech. Rep., Jet Propulsion Laboratory, CALTECH, Pasadena, CA.

Blosch, M., Weiss, S., Scaramuzza, D. and Siegwart, R. (2010). Vision based MAV navigation in unknown and unstructured environments. In: *Proc. IEEE ICRA*, pp. 21–28. Anchorage, Alaska.

Bouabdallah, S., Noth, A. and Siegwart, R. (2004). PID vs LQ control techniques applied to an indoor micro quadrotor. In: *Proc. 1EEE/RSJ Int. Conf. On Intelligent Robots and Systems*, pp. 2451–2456. Sendal, Japan.

Boyd, S., Ghaoui, L.E., Feron, E. and Balakrishnan, V. (1994). *Linear Matrix Inequalities in System and Control Theory*. SIAM, Philadelphia, PA.

Brockett, R.W. and Liberzon, D. (2000). Quantized feedback stabilization of linear systems. *IEEE Trans. Autom. Control*, vol. 45, no. 7, pp. 1279–1289.

Bruckstein, A.M., Sapiro, G. and Shaked, D. (1995). Evolution of planar polygons. *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 9, no. 6, pp. 991–1014.

Bullo, F., Murray, R.M. and Sarti, A. (1995). Control on the sphere and reduced attitude stabilization. In: *IFAC Symposium on Nonlinear Control Systems*. Tahoe City, CA.

Chandler, P.R., Pachter, M. and Rasmussen, S. (2001). UAV cooperative control. In: *Proc. IEEE ACC*, pp. 50 – 55. Arlington, VA.

Chichka, D.F. (2001). Satellite clusters with constant apparent distribution. *AIAA Journal of Guidance, Control, and Dynamics*, vol. 24, no. 1, pp. 117–122.

Choset, H. (2001). Coverage for robotics - a survey of recent results. In: *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126. Kluwer Academic Publishers, Netherlands.

Cortes, J., Martinez, S. and Bullo, F. (2006). Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Trans. Autom. Control*, vol. 51, no. 8, pp. 1289 – 1298.

Davison, E.J. (1966). A method for simplifying linear dynamic systems. *IEEE Trans. Autom. Control*, vol. 11, no. 1, pp. 93–101.

Dimarogonas, D.V., Tsiotras, P. and Kyriakopoulos, K.J. (2009). Leader-follower cooperative attitude control of multiple rigid bodies. *Systems and Control Letters*, vol. 58, pp. 429–435.

Eberly, D.H. (2001). *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*. Academic Press, London, UK.

Egerstedt, M., Hu, X. and Stotsky, A. (2001). Control of mobile platforms using a virtual vehicle approach. *IEEE Trans. Autom. Control*, vol. 46, no. 11, pp. 1777–1782.

Elia, N. and Mitter, S.K. (2001). Stabilization of linear systems with limited information. *IEEE Trans. Autom. Control*, vol. 46, no. 9, pp. 1384–1400.

Faigl, J., Krajnk, T., Vonasek, V. and Preucil, L. (2010). Surveillance planning with localization uncertainty for mobile robots. In: *Proc. 3rd Israeli Conference on Robotics*.

Fax, J.A. (2002). *Optimal and Cooperative Control of Vehicle Formations*. Ph.D. thesis, CALTECH, Pasadena, CA.
Available at: `http://users.cms.caltech.edu/~murray/preprints/fax02-phd.pdf`

Feddema, J.T., Lewis, C. and Schoenwald, D.A. (2002 October). Decentralized control of cooperative robotic vehicles: Theory and application. *IEEE J. Robot. Autom.*, vol. 18, no. 5, pp. 852–864.

Feng, L. (2001). *Analysis, Design, Modeling and Control of Networked Control Systems*. Ph.D. thesis, University of Michigan, Michigan.

Flocchini, P., Prencipe, G., Santoro, N. and Widmayer, P. (2005). Gathering of asynchronous mobile robots with limited visibility. *Theoretical Computer Science*, vol. 337, pp. 147–168.

Fregene, K. (2002). *Distributed Intelligent Control of Hybrid Multiagent Sytems*. Ph.D. thesis, University of Waterloo, Ontario Canada.

Ganguli, A., Cortes, J. and Bullo, F. (2005). On rendezvous for visually-guided agents in a nonconvex polygon. In: *Proc. IEEE CDC*, pp. 5686–5691. Seville, Spain.

Godsil, C. and Royle, G. (2001). *Algebraic Graph Theory*. Springer-Verlag, New York.

Gurdan, D., Stumpf, J., Achtelik, M., Doth, K., Hirzinger, G. and Rus, D. (2007). Energy efficient autonomous four rotor flying robot controlled at 1kHz. In: *Proc. IEEE ICRA*, pp. 361–366. Roma, Italy.

Hernandez, S. and Paley, D.A. (2009). Stabilization of collective motion in a time-invariant flowfield on a rotating sphere. In: *Proc. IEEE ACC.*, pp. 623–628. St. Louis, MO.

Higuchi, K., Shimada, T. and Rekimoto, J. (2011). Flying sports assistant: external visual imagery representation for sports training. In: *Proc. 2nd Augmented Human International Conference*. Megve, France.

Hoffmann, G., Waslander, S. and Tomlin, C. (2008). Quadrotor helicopter trajectory tracking control. In: *Proc. AIAA Guidance, Navigation and Control Conference and Exhibit*. Honolulu, Hawaii.

Horn, R.A. and Johnson, C.R. (1985). *Matrix Analysis*. Cambridge University Press, UK.

Hu, J. (2003). *Multi-Agent Coordination: Theory and Applications*. Ph.D. thesis, University of California, Berkeley, CA.

Hughes, P.C. (2004). *Spacecraft Attitude Dynamics*. Dover Publications Inc, Mineola, NY.

Hwang, I. and Tomlin, C. (2002). Protocol-based conflict resolution for finite information horizon. In: *Proc. IEEE ACC*, pp. 748–753. Anchorage, AK.

Jadbabaie, A., Lin, J. and Morse, A.S. (2003). Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 988–1001.

Jin, Z. (2007). *Study of coupling in interconnected systems*. Ph.D. thesis, CALTECH, Pasadena, CA.

Justh, E.W. and Krishnaprasad, P.S. (2004). Equilibria and steering laws for planar formations. *Systems and Control Letters*, vol. 52, no. 1, pp. 25–38.

Justh, E.W. and Krishnaprasad, P.S. (2005). Natural frames and interacting particles in three dimensions. In: *Proc. Joint 44th IEEE CDC and European control conf.*, pp. 2841–2846. Seville, Spain.

Keviczky, T., Borrelli, F., Fregene, K., Godbole, D. and Balas, G.J. (2008). Decentralized receding horizon control and coordination of autonomous vehicle formations. *IEEE Trans. Control Syst. Technol.*, vol. 16, pp. 19–33.

Kim, Y. and Mesbahi, M. (2004). Quadratically constrained attitude control via semidefinite programming. *IEEE Trans. Autom. Control*, vol. 49, pp. 731–735.

Kim, Y., Mesbahi, M. and Hadaegh, F. (2004). Multiple-spacecraft reconfigurations through collision avoidance, bouncing, stalemate. *Journal of Optimization Theory and Applications*, vol. 122, pp. 323–343.

Kim, Y., Mesbahi, M. and Hadaegh, F.Y. (2003). Dual-spacecraft formation flying in deep space: optimal collision-free reconfigurations. *AIAA Journal of Guidance, Control, and Dynamics*, vol. 25, no. 2, pp. 375–379.

Kim, Y., Mesbahi, M., Singh, G. and Hadaegh, K. (2010). On the convex parameterization of spacecraft orientation in presence of constraints and its applications. *IEEE Trans. Aerosp. Electron. Syst.*, vol. 46, no. 3, pp. 1097–1109.

Klavins, E. and Murray, R.M. (2003). Distributed algorithms for cooperative control. *IEEE Pervasive Computing*, vol. 3, no. 1, pp. 56–65.

Krajnik, T., Vonasek, V., Fiser, D. and Faigl, J. (2011). AR-drone as a platform for robotic research and education. In: *Proc. International Conference on Research and Education in Robotics*. Prague, Czech.

Kuchar, J.K. and Yang, L.C. (2000). A review of conflict detection and resolution modeling methods. *IEEE Trans. Intell. Transp. Syst.*, vol. 1, no. 4, pp. 179–189.

Kuffner, J.J. and LaValle, S.M. (2000). RRT-Connect: An efficient approach to single-query path planning. In: *Proc. IEEE ICRA.*, pp. 995–1001. San Francisco, CA.

Kuipers, J.B. (1999). *Quaternions and Rotation Sequences*. Princeton University Press, Princeton, NJ.

Kuwata, Y. (2003). *Real-time Trajectory Design for Unmanned Aerial Vehicles using Receding Horizon Control*. Master's thesis, MIT, Cambridge, MA.

Kvasnica, M., Grieder, P., Baotic, M. and Morari, M. (2004). *Multi-Parametric Toolbox (MPT)*. Tech. Rep., Automatic Control Laboratory, Swiss Federal Institute of Techonology (ETH), Zurich.

Lafferriere, G., Williams, A., Caughman, J. and Veerman, J. (2005). Decentralized control of vehicle formations. *Systems and Control Letters*, vol. 54, pp. 899–910.

LaValle, S.M. (2006). *Planning Algorithms*. Cambridge University Press.

Lawton, L. and Beard, R.W. (2002). Synchronized multiple spacecraft rotations. *Automatica*, vol. 38, no. 8, pp. 1359–1364.

Leonard, N.E. and Fiorelli, E. (2001). Virtual leaders, artificial potentials and coordinated control of groups. In: *Proc. 40th IEEE Conference on Decision and Control*, vol. 3, pp. 2968–2973.

Leonard, N.E., Paley, D.A., Lekien, F., Sepulchre, R., Fratantoni, D.M. and Davis, R.E. (2007). Collective motion, sensor networks and ocean sampling. *Proc. of the IEEE*, vol. 95, no. 1, pp. 48–74.

Lewis, M.A. and Tan, K. (1997). High precision formation control of mobile robots using virtual structures. *Autonomous Robots*, vol. 4, no. 4, pp. 387–403.

Lin, J., Morse, A.S. and Anderson, B.O. (2003). The multi-agent rendezvous problem. In: *Proc. 13th IFAC World Congress*, pp. 1508–1513.

Lin, Z., Broucke, M.E. and Francis, B.A. (2004). Local control strategies for groups of mobile autonomous agents. *IEEE Trans. Autom. Control*, vol. 49, no. 4, pp. 622–629.

Lin, Z., Francis, B. and Maggiore, M. (2005). Necessary and sufficient graphical conditions for formation control of unicycles. *IEEE Trans. Autom. Control*, vol. 50, no. 1, pp. 121–127.

Liu, X. and Goldsmith, A.J. (2004). Kalman filtering with partial observation losses. In: *Proc. IEEE Conference on Decision and Control*, pp. 4180–4186. Shanghai, China.

Lofberg, J. (2004). Yalmip : A toolbox for modeling and optimization in Matlab. In: *Proc. IEEE CACSD Conference*, pp. 284–289. Taipei, Taiwan.

Lygeros, J. and Godbole, D. (1994). An interface between continuous and discrete event controllers for vehicle automation. *IEEE Trans. on Vehicular Technology*, vol. 46, no. 1, pp. 229–241.

Marshall, J., Broucke, M.E. and Francis, B.A. (2004). Formations of vehicles in cyclic pursuit. *IEEE Trans. Autom. Control*, vol. 49, no. 11, pp. 1963–1974.

MBARI (2006). Monterey bay aquarium research institute.
Available at: `http://www.mbari.org/aosn`

Mehyar, M., Spanos, D., Low, S. and Murray, R. (2005). Distributed averaging on asynchronous communication networks. In: *Proc. 44th IEEE Conference on Decision and Control, and European Control Conference*, pp. 7446–7451.

Mellinger, D., Michael, N. and Kumar, V. (2010). Trajectory generation and control for precise aggressive maneuvers with quadrotors. In: *International Symposium on Experimental Robotics*. New Delhi, India.

Mesbahi, M. and Hadaegh, F.Y. (2001). Formation flying control of multiple spacecraft via graphs, matrix inequalities, and switching. *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 2, pp. 369–377.

Michael, N., Fink, J. and Kumar, V. (2011). Cooperative manipulation and transportation with aerial robots. *Autonomous Robots*, vol. 30, no. 1, pp. 73–86.

Minar, N., Kramer, K.H. and Maes, P. (1999). Cooperating mobile agents for dynamic network routing. In: Hayzelden, A. and Bigham, J. (eds.), *Software Agents for Future Communication Systems*, pp. 287–304. Springer-Verlag, Berlin.

Mitter, S.K. (2000). Control with limited information: the role of systems theory and information theory. *IEEE Information Theory Society Newsletter*, vol. 4, no. 50, pp. 122–131.

Moreau, L. (2003). Leaderless coordination via bidirectional and unidirectional time-dependent communication. In: *Proc. IEEE Conf. Decision and Control*, pp. 3070–3075.

Moreau, L. (2005). Stability of multiagent systems with time-dependent communication links. *IEEE Trans. Autom. Control*, vol. 50, no. 2, pp. 169–182.

Murray, R.M. (2006). Recent research in cooperative control of multi-vehicle systems. *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 571 –583.

Nevala, A.E. (2005). A glide across the gulf stream. *Oceanus*, vol. 44, no. 1.

Ng, W.S. and Sharlin, E. (2011). *Collocated interaction with flying robots*. Tech. Rep., Department of Computer Science, University of Calgary, Calgary, Canada.

Okoloko, I. (2012*a*). Consensus and optimization based collective path planning on a sphere. *Aerospace Science and Technology*, vol. under review.

Okoloko, I. (2012*b*). Path planning for multiple spacecraft using consensus with LMI avoidance constraints. In: *Proc. IEEE Aerospace Conference*. Big Sky, MO.

Okoloko, I. and Basson, A. (2011). Consensus with collision avoidance: An LMI approach. In: *Proc. 5th IEEE International Conference on Automation, Robotics and Applications*, pp. 84–89. Wellington, NZ.

Okoloko, I. and Kim, Y. (2010). Distributed constrained attitude and position control using graph Laplacians. In: *Proc. Third ASME Conference on Dynamic Systems and Control*. Cambridge, MA.

Okoloko, I. and Kim, Y. (2012*a*). Attitude synchronization of multiple spacecraft with cone avoidance constraints. In: *Proc. IEEE Aerospace Conference*. Big Sky, MO.

Okoloko, I. and Kim, Y. (2012*b*). Attitude synchronization of multiple spacecraft with cone avoidance constraints. *IEEE Trans. Aerosp. Electron. Syst.*, vol. under review.

Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Trans. Autom. Control*, vol. 51, no. 3, pp. 401–420.

Olfati-Saber, R. and Murray, R. (2004). Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. Autom. Control*, vol. 49, pp. 1520–1533.

Paganini, F., Doyle, J. and Low, S. (2001). Scalable laws for stable network congestion control. In: *Proc. IEEE Conference on Decision and Control*, pp. 185–190. Orlando, FL.

Paley, D. (2008). Stabilization of collective motion on a sphere. *Automatica*, vol. 41, pp. 212–216.

Pant, A., Seiler, P. and Hedrick, J.K. (2002). Mesh stability of look-ahead interconnected systems. *IEEE Trans. Autom. Control*, vol. 47, no. 2, pp. 403–407.

PATH (2006). California partners for advanced transit and highways.
Available at: `http://www.path.berkeley.edu`

Peaucelle, D., Henrion, D., Labit, Y. and Taitz, K. (2002). *Users Guide for SEDUMI INTERFACE 1.04*. Tech. Rep., LAAS - CNRS, France.

Peng, L., Zhao, Y., Tian, B., Zhang, J., Bing-Hong, W., Hai-Tao, Z. and Zhou, T. (2009). Consensus of self-driven agents with avoidance of collisions. *Physical Review*, vol. E 79.

Ramapriyan, K. (1970). *Study of coupling in interconnected systems*. Ph.D. thesis, University of Minnesota, Minneapolis, Minnesota.

Ren, W. (2004). *Consensus Seeking, Formation Keeping, and Trajectory Tracking in Multiple Vehicle Cooperative Control*. Ph.D. thesis, Brigham Young University, Provo, UT.

Ren, W. (2006). Distributed attitude alignment in spacecraft formation flying. *International Journal of Adaptive Control and Signal Processing*, vol. 21, pp. 95–113.

Ren, W. and Beard, R.W. (2004). Decentralized scheme for spacecraft formation flying via the virtual structure approach. *AIAA Journal of Guidance, Control and Dynamics*, vol. 27, no. 1, pp. 73 –82.

Ren, W. and Beard, R.W. (2005). Consensus seeking in multi-agent systems under dynamically changing interaction topologies. *IEEE Trans. Autom. Control*, vol. 50, no. 5, pp. 655–661.

Ren, W., Beard, R.W. and McLain, T.W. (2005). Coordination variables and consensus building in multiple vehicle systems. In: Kumar, V., Leonard, N.E. and Morse, A.S. (eds.), *Lecture Notes in Control and Information Sciences*, vol. 309, pp. 171–188. Springer-Verlag, Berlin.

Reynolds, C.W. (1987). Flocks, herds, and schools: a distributed behavioral model. *Computer Graphics (ACM SIGGRAPH '87 Conference Proceedings)*, vol. 21, no. 4, pp. 25–34.

Richards, A. (2005). *Robust Constrained Model Predictive Control*. Ph.D. thesis, MIT, Cambridge, CA.

Richards, A., Schouwenaars, T., How, J.P. and Feron, E. (2002). Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming. *AIAA Journal of Guidance Control and Dynamics*, vol. 25, pp. 755–764.

Rodriguez-Angeles, A. and Nijmeijer, H. (2003). Cooperative synchronization of robots via estimated state feedback. In: *Proc. 42nd IEEE Conference on Decision and Control*, vol. 2, pp. 1514–1519. Maui Maui, HI.

Sabol, C., Burns, R. and McLaughlin, C.A. (2001). Satellite formation flying design and evolution. *AIAA Journal of Spacecraft and Rockets*, vol. 38, no. 2, pp. 270–278.

Sandell, N.R., Varaiya, P., Athans, M. and Safonov, M.G. (1978). Survey of decentralized control methods for large scale systems. *IEEE Trans. Autom. Control*, vol. 23, no. 2, pp. 108–128.

Scardovi, L., Leonard, N.E. and Sepulchre, R. (2005). Stabilization of collective motion in three dimensions: A consensus approach. In: *Proc. 46th IEEE conf. decision and control.*, pp. 2931–2936. New Orleans, LA.

Schmidt, M.D. (2011). *Simulation and Control of a Quadrotor Unmanned Aerial Vehicle*. Master's thesis, University of Kentucky, Kentucky, USA.

Schoonderwoerd, R. and Holland, O. (1999). Minimal agents for communications networks routing: The social insect paradigm. In: Hayzelden, A. and Bigham, J. (eds.), *Software Agents for Future Communication Systems*, pp. 305–325. Springer-Verlag, Berlin.

Schwartz, J.T. and Sharir, M. (1983). On the piano movers problem: III. Coordinating the motion of several independent bodies. *International Journal of Robotics Research*, vol. 2, no. 3, pp. 97–140.

Sepulchre, R., Paley, D.A. and Leonard, N.E. (2007). Stabilization of planar collective motion: All-to-all communication. *IEEE Trans. Autom. Control*, vol. 52, no. 5, pp. 811–824.

Sinopoli, B., Schenato, L., Franceschetti, M., Poolla, K., Jordan, M.I. and Sastry, S.S. (2004). Kalman filtering with intermittent observations. *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1453–1464.

Smith, S.L., Broucke, M.E. and Francis, B.A. (2005). A hierarchical cyclic pursuit scheme for vehicle networks. *Automatica*, vol. 41, no. 6, pp. 1045–1053.

Stothert, A. and McLeod, M. (1997). Distributed intelligent control system for a continuous state plant. *IEEE Trans. Syst., Man, Cybern.*, vol. 27, no. 3, pp. 395–401.

Sturm, J.F. (1998). Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, vol. 11-12, pp. 625–653.

Swaroop, D. and Hedrick, J.K. (1996). String stability of interconnected systems. *IEEE Trans. Autom. Control*, vol. 41, no. 3, pp. 349–357.

Tatikonda, S. and Mitter, S.K. (2004). Control under communication constraints. *IEEE Trans. Autom. Control*, vol. 49, no. 7, pp. 1056–1068.

Tiwari, A., Fung, J., Carson-III, J., Bhattacharya, R. and Murray, R.M. (2004). A framework for Lyapunov certificates for multi-vehicle rendezvous problems. In: *Proc. IEEE ACC*, pp. 5582–5587. Boston, MA.

Unwin, S.C. and Beichman, C. (2003). *NASA's Terrestrial Planet Finder*. Tech., Rep., NASA Astrobiology Institute General Meeting, Temple, AZ.
Available at: `http://hdl.handle.net/2014/6376`

van den Berg, J., Guy, S., Lin, M. and Manocha, D. (2009). Reciprocal n-body collision avoidance. In: *Proc. Int. Symposium of Robotics Research - ISRR*. Lucerne, Switzerland.

van den Bergen, G. (2004). *Collision detection in interactive 3D environments*. Morgan Kaufmann Publishers, San Francisco, CA.

Vicsek, T., Czirok, A., Ben-Jacob, E., Cohen, I. and Schochet, O. (1995). Novel type of phase transitions in a system of self-driven particles. *Physical Review Letters*, vol. 75, pp. 1226–1229.

Walsh, G.C., Ye, H. and Bushnell, L.G. (2002). Stability analysis of networked control systems. *IEEE Trans. Control Syst. Technol.*, vol. 10, no. 3, pp. 438–446.

Wen, J. and Kreutz-Delgado, K. (1991). The attitude control problem. *IEEE Trans. Autom. Control*, vol. 36, pp. 1148–1161.

Wong, S.W. and Brockett, R.W. (1997). Systems with finite communication bandwidth constraints - part I: State estimation problems. *IEEE Trans. Autom. Control*, vol. 42, no. 9, pp. 1294–1299.

Xiao, L., Boyd, S. and Lall, S. (2005). A scheme for robust distributed sensor fusion based on average consensus. In: *IEEE International Conf. on Information Processing in Sensor Networks*, pp. 63–70.

Yamaguchi, H., Arai, T. and Beni, G. (2001). A distributed control scheme for multiple robotic vehicles to make group formations. *Robotics and Autonomous Systems*, vol. 36, pp. 125–147.

Yanakiev, D. and Kanellakopoulos, I. (1996). A simplified framework for string stability analysis in AHS. In: *Proc. 13th IFAC World Congress*, vol. Q, pp. 66–95.

Yeh, H., Nelson, E. and Sparks, A. (2002). Nonlinear tracking control for satellite formations. *AIAA Journal of Guidance, Control, and Dynamics*, vol. 25, no. 2, pp. 376–386.

Zickler, S., Bruce, J., Biswas, J., Licitra, M. and Veloso, M. (2009). CMDragons extended team description. In: *Proc. CD of RoboCup 2009: Robot Soccer World Cup XIII*.