COMBINED FORMATION CONTROL AND PATH PLANNING FOR MOBILE ROBOT NETWORKS

by

Semih Cemil Dumlu

B.S., Electrical Engineering, Istanbul Technical University, 2007

B.S., Electronics Engineering, Istanbul Technical University, 2008

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Department of Electrical and Electronics Engineering

Boğaziçi University

2011

COMBINED FORMATION CONTROL AND PATH PLANNING FOR MOBILE ROBOT NETWORKS

APPROVED BY:

Assoc. Prof. Mehmet Akar (Thesis Supervisor)	.....
Assoc. Prof. Yağmur Denizhan	.....
Assist. Prof. Çetin Yılmaz	.....

DATE OF APPROVAL: 26.09.2011

ACKNOWLEDGEMENTS

This thesis would not have been possible if my supervisor Assoc. Prof. Mehmet Akar had not supported me. I would like to express my profound gratitude to him for his trust, support, supervision and leadership.

I wish to express my sincere gratitude to my family who made their support available at all times during my study.

Finally, I thank TÜBİTAK for their financial support throughout my research.

ABSTRACT

COMBINED FORMATION CONTROL AND PATH PLANNING FOR MOBILE ROBOT NETWORKS

In this thesis, formation control and path planning algorithms in mobile robot networks are examined for both fixed and dynamically changing interaction topologies. The system is modelled using graph-based framework where the nodes denote the agents and the communication links are represented as edges of the graph. Based on this graph model, well-established algebraic graph theoretic results are employed to examine graph-based formation control for point mass holonomic and real omnidirectional robotic networks. In addition, path planning using potential fields that enables a leader as well as followers avoid obstacles in the environment while maintaining formation is analyzed. Subsequently, performance of the combined path planning and distributed formation control algorithms including saturated velocity control inputs is examined by the help of numerical simulations and it is verified that all followers reach and maintain the formation while avoiding obstacles.

ÖZET

GEZGİN ROBOT AĞLARINDA BİRLEŞİK DİZİLİM DENETİMİ VE YOL PLANLAMASI

Bu tezde, gezgin robot ağlarında dizilim denetimi ve yol planlama algoritmaları hem sabit hem devingen değişken etkileşimli ilingelerde incelenmiştir. Sistem çizge tabanlı çerçeve kullanılarak düğümler aracıları, kenarlar ise iletişim bağlantılarını temsil etmek kaydıyla modellenmiştir. Bu çizge modeline dayanarak yerleşik cebirsel çizge teori sonuçları, çizge tabanlı dizilim denetimini noktasal kütle holonom ve gerçek her yönlü robotik ağlarında incelemek için kullanılmıştır. Ek olarak, lider ve takipçilerin çevredeki engellerden dizilimi koruyarak kaçınmalarını sağlayan potansiyel alan yol planlaması ele alınmıştır. Akabinde, doymuş hız denetimi girişlerini içeren birleşik yol planlaması ve dağıtılmış dizilim denetimi algoritmalarının başarımı sayısal benzetimler yardımıyla incelenmiştir. Bu çalışmalar doğrulamıştır ki tüm takipçiler dizilime ulaşmış ve bu dizilimi engellerden kaçınarak sürdürmüştür.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES ix
LIST OF TABLES xiii
1. INTRODUCTION 1
1.1. Organization of the Thesis 2
2. OVERVIEW OF CONSENSUS ALGORITHMS IN MOBILE ROBOT NETWORKS 4
2.1. Introduction 4
2.2. Overview of Graph Theory 5
2.3. Fundamentals of Consensus Algorithms 9
2.4. Consensus Under Fixed Interaction Topologies 10
2.4.1. Convergence Analysis of Consensus Algorithms in Continuous-Time 10
2.4.2. Numerical Examples for Fixed Interaction Topologies 12
2.5. Consensus Under Dynamically Changing Interaction Topologies 14
2.5.1. Convergence Analysis of Consensus Algorithms in Continuous-Time 14
2.5.2. Numerical Examples for Dynamically Changing Interaction Topologies 15
2.6. Connectivity Analysis of Consensus Algorithms 15
2.6.1. Numerical Examples for the Connectivity of Consensus Algorithms 16
2.6.2. Future Research of Connectivity Analysis of Consensus Algorithms 18
2.7. Summary of the Chapter 18
3. GRAPH-BASED FORMATION CONTROL FOR MOBILE ROBOT NETWORKS 19
3.1. Graph-Based Formation Control 21
3.1.1. Leader Based Multi-Agent Control 23

3.2. Convergence Analysis 24
3.3. Convergence Speed Analysis 26
3.3.1. Numerical Examples for Performance Analysis of Consensus Algorithms 27
3.4. Formation Control Simulations and Results 28
3.4.1. Point Mass Simulations of Holonomic Robotic Agents 28
3.4.1.1. Distributed Consensus Without a Leader Case 28
3.4.1.2. Leader-Based Distributed Formation 30
3.4.2. Dynamic Simulation of Omni-Wheeled Holonomic Robotic Agents 32
3.4.2.1. Modelling of a Four Wheeled Omnidirectional Robot . 32
3.4.2.2. Webots: A Simulation Environment 35
3.4.2.3. Distributed Consensus Without a Leader Case 36
3.4.2.4. Leader Based Distributed Formation Case 37
3.5. Summary of the Chapter 38
4. MOTION PLANNING 39
4.1. Overview of Concepts in Motion Planning 39
4.1.1. Properties of the Robot 40
4.1.2. Properties of the Algorithm 40
4.2. Configuration Space 41
4.3. Potential Field Path Planning 41
4.3.1. Attractive Potential 43
4.3.2. Repulsive Potential 44
4.4. Road Map Path Planning 45
4.4.1. Visibility Graph 45
4.4.2. Voronoi Diagram 46
4.5. Cell Decomposition 47
4.5.1. Exact Cell Decomposition 48
4.5.2. Approximate Cell Decomposition 48
4.6. Sampling-Based Algorithms 51
4.6.1. The Probabilistic Road Map Method (PRM) 51
4.6.2. Rapidly Exploring Dense Trees (RDT) 51

4.7. Combined Formation Control and Path Planning 52
4.7.1. Formation Control and Potential Field Path Planning 52
4.8. Summary of the Chapter 55
5. PERFORMANCE ANALYSIS OF CONSENSUS AND DISTRIBUTED FORMATION CONTROL ALGORITHMS 56
5.1. Performance Analysis of Consensus Algorithm 56
5.1.1. Proportional Control 57
5.1.2. Saturated Control 58
5.1.3. Time-Improved Control 59
5.1.4. Maximum Velocity Control 60
5.2. Performance Analysis of Distributed Formation Control Algorithm . . . 61
5.2.1. Proportional Control 62
5.2.2. Saturated Control 62
5.2.3. Time-Improved Control 63
5.2.4. Maximum Velocity Control 63
5.3. Summary of the Chapter 65
6. CONCLUSION 67
REFERENCES 68

LIST OF FIGURES

Figure 2.1. The graph representation of a four node network. 7

Figure 2.2. Gershgorin Theorem applied to Graph Laplacian [8]. 8

Figure 2.3. A strongly connected and balanced graph of three nodes. 12

Figure 2.4. Average consensus for a strongly connected and balanced graph. . 13

Figure 2.5. A directed graph of three nodes with a directed spanning tree. . . 13

Figure 2.6. Weighted average consensus for a graph containing a directed spanning tree. 14

Figure 2.7. Switching between two graphs that the union of the directed graphs (a) and (b) has a directed spanning tree. 15

Figure 2.8. Weighted average consensus for a dynamically changing interaction topology. 16

Figure 2.9. A directed graph with 2 strongly connected components. 17

Figure 2.10. Average consensus for a graph with two strongly connected components. 17

Figure 3.1. Block diagram of the collective networked system (reprinted from [7]). 22

Figure 3.2. Gershgorin circles of the Perron matrices. 25

Figure 3.3. Graph representation of the network placed in the initial positions. 29

Figure 3.4. Agents reaching an average consensus. 29

Figure 3.5. Target relative positions of the rhombus formation. 30

Figure 3.6. Agents reaching a formation. 30

Figure 3.7. Linear trajectory. 31

Figure 3.8. Circular trajectory. 31

Figure 3.9. Technical drawing of Four Wheeled Robot Wheelbase [40]. 32

Figure 3.10. Locomotion System: Omnidirectional wheel [40]. 33

Figure 3.11. Arrangement of n omnidirectional wheels [41]. 33

Figure 3.12. Webots development stages [45]. 35

Figure 3.13. Webots environment of the study. 36

Figure 3.14. Consensus Simulation in (a) Matlab (b) Webots. 37

Figure 3.15. Formation Simulation in (a) Matlab (b) Webots. 37

Figure 3.16. Linear Trajectory Simulation in (a) Matlab (b) Webots. 38

Figure 4.1. Typical potential field generated by the attracting goal and three obstacles. (a) Configuration (b) Equipotential plot (c) Resulting potential field [46]. 42

Figure 4.2. Visibility graph [46]. 46

Figure 4.3. Voronoi diagram [46]. 47

Figure 4.4. Exact cell decomposition example [49]. 48

Figure 4.5. Grid based representation example [49]. 49

Figure 4.6. NF1 function example. S, start; G, goal [49]. 49

Figure 4.7. Approximate variable-cell decomposition example [49]. 50

Figure 4.8. Initial positions and rhombus formation of all agents. 53

Figure 4.9. Leader based distributed formation control with obstacle avoidance and path planning. 53

Figure 4.10. Velocity graph of agents. 54

Figure 4.11. Relative distance graph between leader and followers. 54

Figure 5.1. Graph representation of the network placed in the initial positions. 57

Figure 5.2. Simulation for Proportional Control with $T_{exe} = 55\,s$. 58

Figure 5.3. Simulations for Saturated Control with (a) $v_{max} = 0.05\,m/s$, $T_{exe} = 90\,s$ (b) $v_{max} = 0.2\,m/s$, $T_{exe} = 58\,s$ and (c) $v_{max} = 0.7\,m/s$, $T_{exe} = 55\,s$. 59

Figure 5.4. Simulation for Time-Improved Control with $T_{exe} = 18.2793\,s$ with $k_3 = 55$. 60

Figure 5.5. Simulation for Maximum Velocity Control with $T_{exe} = 28\,s$. 60

Figure 5.6. Target relative positions of the rhombus formation. 61

Figure 5.7. Simulation for Proportional Control with $T_{exe} = 54\,s$. 62

Figure 5.8. Simulations for Saturated Control with (a) $v_{max} = 0.05\,m/s$, $T_{exe} = 80\,s$ (b) $v_{max} = 0.2\,m/s$, $T_{exe} = 55\,s$ and (c) $v_{max} = 0.7\,m/s$, $T_{exe} = 54\,s$. 63

Figure 5.9. Simulation for Time-Improved Control with $T_{exe} = 15.1377\,s$ with $k_3 = 54$. 64

Figure 5.10. Simulation for Maximum Velocity Control with $T_{exe} = 25\,s$. 64

LIST OF TABLES

Table 5.1. Performance Evaluation of the Proposed Algorithms. 66

1. INTRODUCTION

Since the pioneering work by Reynolds [1] on simulation of a flock of birds in flight using a model based on simple local interaction rules, a vast amount of research has been carried out on formation control of mobile robot networks. A mobile robot network is any system in which two or more autonomous robots interact with each other in order to achieve a certain objective such as maintaining formation in addition to performing other tasks. A simple and practical example would be multiple unmanned aerial vehicles (UAVs) assigned to a certain target that can reduce the mission cost and improve the system robustness compared to a single one. In this sense, the advantages of using formation in mobile robot networks are the increase of feasibility, accuracy, robustness, flexibility and efficiency on one side and the decrease of cost and energy on the other side.

In the literature, a wide range of strategies and different approaches have been proposed for formation control of mobile robot networks. One can classify these approaches into three main categories: leader following, behavioral and virtual structure approach. Clearly, each approach has its own advantages and disadvantages [2–4].

In the leader-following approach, some robots are selected as leaders while the rest of the robots are just followers. The main idea is that the followers track the position and orientation of the leader with some prescribed offset. Around this main idea, several variations of this approach have been considered, for example acting like multiple leaders, forming a chain and other tree topologies. Being easy to understand and to implement is one of the advantages of this approach. Moreover, even if the leader is perturbed by some disturbances, the formation can still be maintained. However, having no explicit feedback from the followers to the leader is a disadvantage of this scheme.

The basic idea behind the behavioral approach is to prescribe several desired behaviors such as collision/obstacle avoidance, goal/target seeking and formation keeping

for each robot and calculate the formation control from a weighting of the relative importance of each behavior. The advantage of applying this approach is that it is natural to derive control strategies when mobile robots have multiple competing objectives and furthermore an explicit feedback is included through communication between neighbors. The disadvantages of this approach are that the group behavior cannot be explicitly defined, and it is difficult to analyze the approach mathematically and guarantee the group stability.

In the virtual structure approach, the entire formation is considered as a single structure or rigid body. When the structure moves, it traces out desired trajectories for each robot in the group to track. This approach provides a fairly easy way to prescribe the coordinated behavior of the group and a very good maintenance of the formation during maneuvers. On the other hand, the request of the formation to act as a virtual structure limits the class of potential applications like the cases where the formation shape is time-varying or needs to be frequently reconfigured. In such cases, the virtual structure approach may not be the optimal choice indeed.

Graph-theoretic models can be used to model the problem of formation in multiple mobile robot systems in order to describe the local interactions in the formation. In these models, nodes symbolize the physical entities (agents) and edges represent virtual entities that support the information flow between the nodes. Most notably of recent results that have emerged in the general area of graph-based multi-agent control might be the one on consensus. With the help of averaging based consensus idea, it has become possible to drive scalar states to a common value for different robots, in a completely decentralized fashion.

1.1. Organization of the Thesis

In Chapter 2, first an overview of graph theory is given to see the relationship between consensus algorithms and networks. Subsequently, the consensus algorithm is studied in both fixed and dynamically changing interaction topologies and convergence analysis has been carried out in continuous time illustrated by numerical examples.

In Chapter 3, graph-based formation control is discussed and the performance of the relative position based distributed formation control algorithm on two types of systems, a point mass holonomic robotic agent system and dynamically simulated omnidirectional robotic agent system with and without a leader is examined. The latter system has also been realized using Webots Environment.

In Chapter 4, an overview on motion planning is given. Furthermore, the combination of potential field path planning with leader based distribution formation control is investigated along with a convergence analysis has been carried.

In Chapter 5, performances of consensus and distributed formation control algorithms for different control inputs have been evaluated in terms of execution time and final states.

Finally, concluding remarks and further research ideas are given in Chapter 6.

2. OVERVIEW OF CONSENSUS ALGORITHMS IN MOBILE ROBOT NETWORKS

This chapter overviews consensus algorithms in mobile robot networks. The motivation for information consensus and a literature survey on consensus algorithms are given. Theoretical results regarding consensus under both fixed and dynamically changing interaction topologies are summarized. A few specific examples of consensus algorithms are provided with their simulation result.

2.1. Introduction

In networks of agents, the most common consensus definition [3, 6, 7], is to reach an agreement regarding a certain quantity of interest that depends on the state of all agents. In this regard, a consensus algorithm is an interaction rule that specifies the information exchange between an agent and all of its neighbours in the network.

In consensus problems, the common variable of interest is the information state that represents an instantiation of coordination. Among examples of consensus are local representations of the center and shape of a formation, the rendezvous time, the length of a perimeter being monitored, the direction of motion for a swarm, and the probability that a military target has been destroyed. Assuming only neighbor-to-neighbor interaction between robots, consensus algorithms are, by necessity, designed to be distributed i.e. the update of robots' information state values is based on the information states of their neighbors. The aim is to derive an update law such that the information states of all of the robots in the network converge to a common value.

Consensus algorithms have a historical perspective in [10, 11], and have recently been studied extensively in the context of cooperative control of multiple mobile robots [8,9,12–14]. There are applications of consensus algorithms in different type of problems such as rendezvous [16,17,19,20], formation control [5,9,21], flocking [22,23], perimeter

monitoring [24] and sensor networks [25,26]. In order to understand theoretical results regarding consensus algorithms better, a formal definition of graphs and related topics should be given.

2.2. Overview of Graph Theory

Definition 2.1. *A* graph *is an ordered pair* $G = (V, E)$ *constituted by a set* $V = \{v_1, v_2, ..., v_n\}$ *of vertices or nodes together with a set* $E \subseteq V \times V$ *of edges or lines, which are two-element subsets of* V *which means that an edge is related with two vertices, and the relation is represented as unordered pair of the vertices with respect to the particular edge.*

Graphs have different types depending on their particularities. It can be undirected or directed depending the type of edges. If all edges have no arrows/direction then the graph is said to be undirected otherwise it is directed. Below definitions of some special graphs are given.

Definition 2.2. *A graph* G *is said to be* symmetric *if and only if an edge* $(i, j) \in E$ *implies edge* $(j, i) \in E$.

Definition 2.3. *A graph is said to be* strongly connected *if there exists a path constituted of edges (with the correct direction) between any pair of nodes.*

There are two ways of representing graphs: (i) the geometric way where nodes are represented usually by circles, and edges by line segments with arrows if directed and (ii) the algebraic way using different matrices like adjencency matrix or Laplacian matrix.

Definition 2.4. *An* adjacency matrix *denoted* $A = [a_{ij}]$ *of graph* G *is an* $n \times n$ *matrix where*

$$a_{ij} = \begin{cases} 1 & \textit{if there is communication from node } j \textit{ to node } i \\ 0 & \textit{otherwise} \end{cases}$$

Definition 2.5. *The* in-degree *and* out-degree *of node v_i in a directed graph are defined as*

$$deg_{in}(v_i) = \sum_{j=1}^{n} a_{ji}, \qquad deg_{out}(v_i) = \sum_{j=1}^{n} a_{ij} \tag{2.1}$$

Definition 2.6. Degree matrix *of a directed graph G is an $n \times n$ diagonal matrix Δ where $\Delta_{ij} = 0$ for all $i \neq j$ and $\Delta_{ij} = deg_{out}(v_i)$ for all $i = j$.*

Definition 2.7. *For a directed graph with* $0-1$ *adjacency elements, the graph* Laplacian *is defined as:*

$$L(G) = \Delta - A \tag{2.2}$$

It can be shown that

$$l_{ij} = \begin{cases} \sum_{k=1,k\neq i}^{n} a_{ik} & \text{if } j = i \\ -a_{ij} & \text{if } j \neq i \end{cases} \tag{2.3}$$

From the above definitions one can state that the sum of the values of each row of the Laplacian matrix equals to zero. Therefore, there exists at least one eigenvalue with a zero value. This corresponds to a right eigenvector $w_r = [1, 1, ..., 1]^T$ with identical nonzero elements. This means that the rank of the Laplacian is $rank(L) \leq n - 1$. In the case of strongly connected graphs, we have the following results.

Theorem 2.1. *If a directed graph G of n nodes with a weighted adjacency matrix A is strongly connected, then the rank of its corresponding Laplacian matrix is rank(L) = n - 1, [8].*

However, the converse of the statement in Theorem 2.1 does not hold. In other words, a Laplacian with $rank(L) = n - 1$ does not necessarily imply the strong connectivity of a graph.

Example 2.1. The Laplacian matrix corresponding to the network in Figure 2.1 is as

follows. The rank of the Laplacian is $r(L) = 3$, where $n = 4$ but as in Figure 2.1 the represented graph is not strongly connected since there exists no path from node 4 to any other node.

$$L = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & -1 & 2 \end{bmatrix}$$

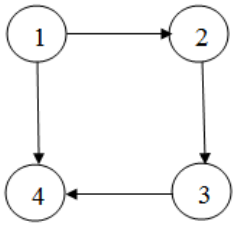

Figure 2.1. The graph representation of a four node network.

Theorem 2.2. *Every eigenvalue of an $n \times n$ matrix A, lies within at least one of the Gershgorin circles defined by:*

$$D(a_{ii}, R_i) = \left\{ z \in C : |z - a_{ii}| \leq \sum_{j=1, j \neq i}^{n} |a_{ij}| \right\} \tag{2.4}$$

where a_{ii} is the ith diagonal element of A, [8].

Applying Gershgorin circle theorem to the Laplacian L of the directed graph G, all the eigenvalues of L are located in the union of the following n disks:

$$D_i = \left\{ z \in C : |z - l_{ii}| \leq \sum_{j=1, j \neq i}^{n} |l_{ij}| \right\} \tag{2.5}$$

It must be noted that for the directed graph G, $l_{ii} = \Delta_{ii}$, and $\sum_{j=1, j \neq i}^{n} |l_{ij}| = deg_{out}(v_i) = \Delta_{ii}$. It follows from Theorem 2.2 and the definitions above that

$$D_i = \{ z \in C : |z - \Delta_{ii}| \leq \Delta_{ij} | \} \tag{2.6}$$

All these n disks are contained in the largest disk $D(G)$ with radius $d_{max}(G)$. As shown in Figure 2.2, the eigenvalues of the matrix $-L$ are located in the disk

$$D_G^{'} = \{z \in C : |z + d_{max}(G)| \leq d_{max}(G)|\} \tag{2.7}$$

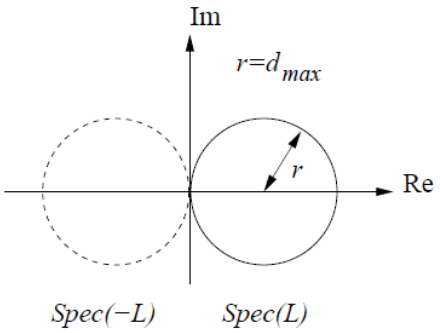

Figure 2.2. Gershgorin Theorem applied to Graph Laplacian [8].

Definition 2.8. *A node v_i of a directed graph G is* balanced *if and only if its in-degree and out-degree are equal, i.e. $deg_{out}(v_i) = deg_{in}(v_i)$. By the same definition, a graph is said to be balanced if and only if all of its nodes are balanced. It can also be stated as, for all i,*

$$\sum_j a_{ij} = \sum_j a_{ji} \tag{2.8}$$

Theorem 2.3. *Given a network G with fixed topology and an adjacency matrix A, the network is balanced if and only if $w_l = e$, that is, the unit vector $e = [1, 1, ..., 1]^T$ is the left eigenvector of the Laplacian L of G, i.e. $e^T L = 0$, [8].*

The proof of the above result proceeds as follows [8]. $\Delta_{ii} = deg_{out}(v_i)$ and $deg_{in}(v_i) = \sum_{j,j\neq i} a_{ji}$, thus the ith column sum of L is equal to zero, or $\sum_j l_{ji} = \sum_{i,j\neq i} l_{ji} + l_{ii} = -deg_{in}(v_i) + deg_{out}(v_i) = 0$, if and only if node v_i of G is balanced. Noting that the i column sum of L equals to the ith element of the row vector $e^T L$, it is concluded that $e^T L = 0$ if and only if all the nodes of G are balanced, that is G is balanced.

2.3. Fundamentals of Consensus Algorithms

Suppose that there are n agents in a team. The directed graph $G = (V_n, E_n)$, where $V_n = \{1, ..., n\}$ is the node set and $E_n \subseteq V_n \times V_n$ is the edge set, is used to model the interaction topology among the n agents. Let $A_n = [a_{ij}] \in \mathbb{R}^{n\times n}$ and $L_n = [l_{ij}] \in \mathbb{R}^{n\times n}$ be, respectively, the adjacency matrix and the non-symmetrical Laplacian matrix associated with G_n. Also an undirected graph can be viewed as a special case of a directed graph. A directed graph takes into account the general case where information flow may be unidirectional.

Consider information states with single-integrator dynamics given by

$$\dot{x_i} = u_i, i = 1, ..., n \tag{2.9}$$

where $x_i \in \mathbb{R}^m$ is the information state and $u_i \in \mathbb{R}^m$ is the information control input of the ith agent. A continuous-time consensus algorithm is given by

$$u_i = -\sum_{j=1}^{n} a_{ij}(x_i - x_j), i = 1, ..., n, \tag{2.10}$$

where $a_{ij}(t)$ is the (i, j) entry of the adjecency matrix $A_n(t) \in \mathbb{R}^{n\times n}$ at time t [8]. Note that $a_{ij} > 0$ if $(j, i) \in E_n$ at time t and $a_{ij}(t) = 0$ otherwise, $\forall j \neq i$. Equation 2.9 and 2.10 can be rewritten in matrix form as

$$\dot{x} = -L_n(t)x \tag{2.11}$$

where $x = [x_1^T, ..., x_n^T]^T, L_n(t) \in \mathbb{R}^{n\times n}$ is the non symmetrical Laplacian matrix at time t. With Equation 2.10, consensus is said to be achieved or reached by the team of agents if, for all $x_i(0)$ and all $i, j = 1, ..., n, \|x_i(t) - x_j(t)\| \to 0, as\, t \to \infty$.

Note that the continuous-time consensus algorithm in Equation 2.10 is distributed in the sense that each agent needs information only from its neighbors. Also note that

the interaction topology may be changing dynamically due to unreliable transmission or a limited communication range. $G_n(t)$ is used to denote the dynamically changing interaction topologies corresponding to Equation 2.10. For a given interaction topology, the weights $a_{ij}(t)$ in Equation 2.10 may be time-varying to represent possibly the time-varying relative confidence of each agents information state or relative reliabilities of different interaction links among agents. As a result, $L_n(t)$ in Equation 2.11 may be dynamically changing over time.

2.4. Consensus Under Fixed Interaction Topologies

2.4.1. Convergence Analysis of Consensus Algorithms in Continuous-Time

In this section, the conditions under which the states of the consensus algorithm presented in Equation 2.10 converge when the communication topology is fixed, that is, the Laplacian matrix L is constant, will be investigated. Necessary and sufficient conditions for consensus of information using consensus algorithms in continuous-time will be derived.

If the directed communication topology is both strongly connected and balanced, then $e = [1, 1, ..., 1]^T$ is a left eigenvector of L associated with the simple zero eigenvalue. Therefore, as shown in [8], average consensus is achieved if and only if the directed communication topology is both strongly connected and balanced. It can be shown that, in the case of undirected communication, average consensus is achieved if and only if the topology is connected.

In contrast, when the directed communication topology contains a rooted directed spanning tree, the consensus equilibrium is equal to the weighted average of the initial conditions of those agents that have a directed path to all of the other vehicles as shown in [15, 38, 39]. Requiring a rooted directed spanning tree is less stringent than requiring a strongly connected and balanced graph.

In order to derive the main results on consensus, we need first to enumerate some

properties that will be useful later.

Lemma 2.1. *Given a matrix* $A = [a_{ij}] \in \mathbb{R}^{n\times n}$, *where* $a_{ii} \leq 0, a_{ij} \geq 0, \forall i \neq j$, *and* $\sum_{j=1}^{n} a_{ij} = 0$ *for each* i, *then* A *has at least one zero eigenvalue with an associated eigenvector* e_n, *and all nonzero are in the open left half plane. Furthermore,* A *has exactly one zero eigenvalue if and only if the directed graph of* A, *denoted by* $\gamma(A)$, *has a directed spanning tree, [3].*

Lemma 2.2. *If* L_n *is the* $n \times n$ *nonsymmetrical Laplacian matric, then* $e^{-L_n t}, \forall t \geq 0$, *is a row-stochastic matrix with positive diagonal entries. In addition,* $rank(L_n) = n - 1$ *if and only if* L_n *has a simple zero eigenvalue. Furthermore, if* L_n *has a simple zero eigenvalue and* $v = [v_1 ... v_n]^T \geq 0$ *satisfy* $e_n^T v = 1$ *and* $L_n^T v = 0$, *then* $e^{-L_n t} \to e_n^T v T, as\, t \to \infty$, *[3].*

The following theorem shows a necessary and sufficient condition for fixed interaction topologies and constant gains a_{ij}.

Theorem 2.4. *Suppose that* A_n *is constant. Equation 2.10 achieves consensus asymptotically if and only if directed graph* G_n *has a directed spanning tree. In particular,* $x_i(t) \to \sum_{i=1}^{n} v_i x_i(0)$, *as* $t \to \infty$, *where* $v = [v_1, ..., v_n]^T \geq 0, e_n T v = 1$, *and* $L_n^T v = 0$, *[3].*

Proof. (Sufficiency) $e^{-L_n t} \to e_n c^T$, where c is an $n \times 1$ vector, is needed to be shown. Obviously $-L_n$ in (2.11) associated with G_n has the same properties as A in Lemma 2.1. The fact that G_n has a directed spanning tree implies that the directed graph of $-L_n$ has also a directed spanning tree. Therefore, we know from Lemma 2.1 that $-L_n$ has exactly one zero eigenvalue and all other eigenvalues are in the open left half plane. As a result, we know that the continuous-time algorithm in Equation 2.10 achieves consensus asymptotically according to Lemma 2.2. With 2.10 the solution of 2.9 is given by $x(t) = e^{-L_n t} x(0)$. Therefore, it follows from Lemma 2.2 that

$$x_i(t) \to \sum_{i=1}^{n} v_i x_i(0), i = 1, ..., n, as\, t \to \infty. \tag{2.12}$$

(Necessity) Suppose that algorithm given in Equation 2.10 achieves consensus asymptotically but that G_n does not have a directed spanning tree. Then there exist at least two agents i and j such that there is no path in G_n that contains both i and j. Therefore it is impossible to bring data between these two vehicles into consensus, which implies that consensus cannot be achieved asymptotically, [3]. □

2.4.2. Numerical Examples for Fixed Interaction Topologies

Example 2.2. Consider a network of three nodes represented by the strongly connected and balanced directed graph G shown in Figure 2.3 with the following adjacency matrix A

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}, L = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{bmatrix}$$

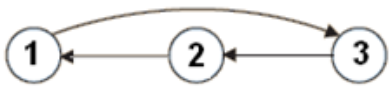

Figure 2.3. A strongly connected and balanced graph of three nodes.

Figure 2.4 shows the information values of the nodes in the network reaching a consensus value, 0.4, which is the average of the initial states $[0.2, 0.4, 0.6]^T$.

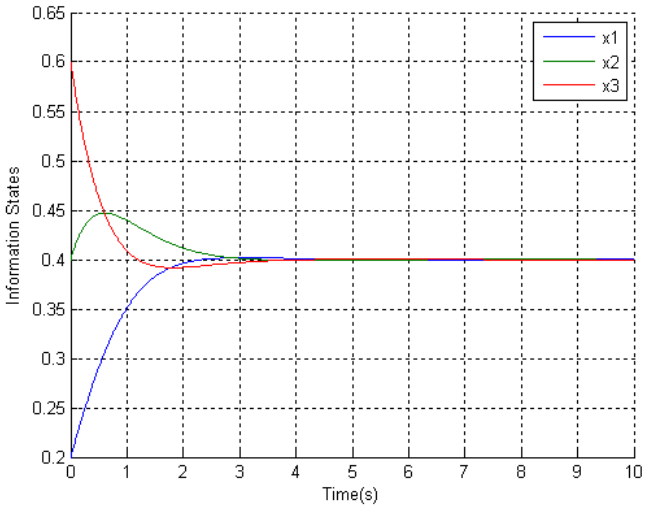

Figure 2.4. Average consensus for a strongly connected and balanced graph.

Example 2.3. Consider a network of three nodes represented by a graph G which has a directed spanning tree like in Figure 2.5 with the following adjacency matrix A.

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, L = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$



Figure 2.5. A directed graph of three nodes with a directed spanning tree.

Figure 2.6 shows the information states of the nodes in the network reaching a consensus value, 0.6, which is the initial state of node 3.

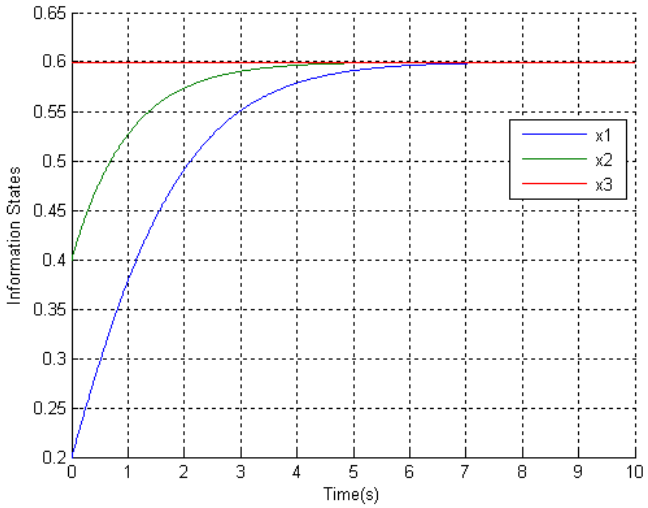

Figure 2.6. Weighted average consensus for a graph containing a directed spanning tree.

2.5. Consensus Under Dynamically Changing Interaction Topologies

2.5.1. Convergence Analysis of Consensus Algorithms in Continuous-Time

In this section, it is assumed that the interaction topology is dynamically changing and $L_n(t)$ is time-dependent. Necessary and sufficient conditions for consensus of information using consensus algorithms in continuous-time are derived. The following theorem for Equation 2.10 under dynamically changing interaction topologies is stated below. The proof of Theorem 2.5 is stated in [3].

Theorem 2.5. *Let $t_1, t_2, ...$ be the infinite time sequence such that $\tau_i = t_{i+1} - t_i \in Y, i = 0, 1, ...$ Let $G_n(t_i) \in \overline{G_n}$ be the directed graph at time $t = t_i$ and $a_{ij}(ti) \in \overline{\Psi}$, where $\overline{\Psi}$ is a finite set of arbitrary nonnegative numbers. Continuous-time algorithm in Equation 2.10 achieves consensus asymptotically if there exists an infinite sequence of continuous, nonempty,, uniformly bounded, time intervals $(t_{i_j}, t_{i_{j+1}}), j = 1, 2, ...,$ starting at $t_{i_1} = t_0$, with property that the union of the directed graphs across each such interval has a directed spanning tree. Furthermore, if the union of the directed graphs after some finite time does not have a directed spanning tree, then consensus cannot be achieved asymptotically, [3].*

2.5.2. Numerical Examples for Dynamically Changing Interaction Topologies

Example 2.4. A network with three nodes is simulated by switching the two graphs that the union of the directed graphs has a directed spanning tree shown in Figure 2.7 with the following adjacency matrices A_1 and A_2. Figure 2.8 shows that the information states of the nodes in the network reach a consensus value, 0.6, which is the initial state of node 3.

$$A_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, L_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, L_2 = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Figure 2.7. Switching between two graphs that the union of the directed graphs (a) and (b) has a directed spanning tree.

2.6. Connectivity Analysis of Consensus Algorithms

The following lemma combines a well-known rank property of graph Laplacians with Gershgorin theorem to provide spectral characterization of Laplacian of a fixed directed network G. Before stating the lemma, note that the notion of strong connectivity of graphs is defined in Definition 2.3.

Lemma 2.3. *Let G be a strongly connected digraph on n nodes. Then $rank(L) = n-1$ and all nontrivial eigenvalues of L have positive real parts. Furthermore, suppose G has $c \geq 1$ strongly connected components, then $rank(L) = n - c$, [7].*

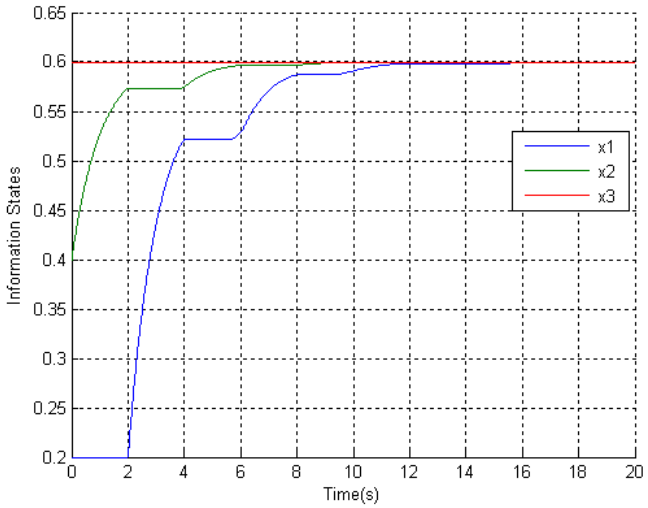

Figure 2.8. Weighted average consensus for a dynamically changing interaction topology.

Remark 2.1. *Lemma 2.3 holds under a weaker condition of existence of a directed spanning tree for G. G has a directed spanning tree if there exists a node r (a root) such that all other nodes can be linked to r via a directed path. This type of condition on existence of directed spanning trees has appeared in [12–14]. The root node is commonly known as a leader, [7, 12]*

2.6.1. Numerical Examples for the Connectivity of Consensus Algorithms

Example 2.5. Consider a network of five nodes represented by graph G which has 2 strongly connected components (1-3-4, 2-5) shown in Figure 2.9 with the following adjacency matrix A and L. Figure 2.10 shows that the information states of the nodes in the network reach a consensus values, 0.533 and 0.7, which is the average of the initial states of node 1-3-4 and 2-5 respectively as expected.

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}, L = \begin{bmatrix} 2 & 0 & -1 & -1 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \end{bmatrix}$$

$$rank(L) = n - c = 5 - 2 = 3$$

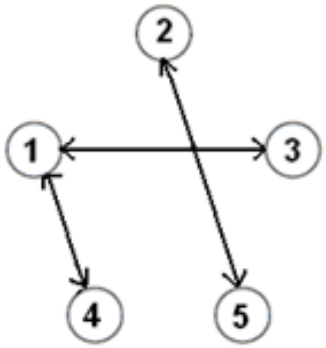

Figure 2.9. A directed graph with 2 strongly connected components.

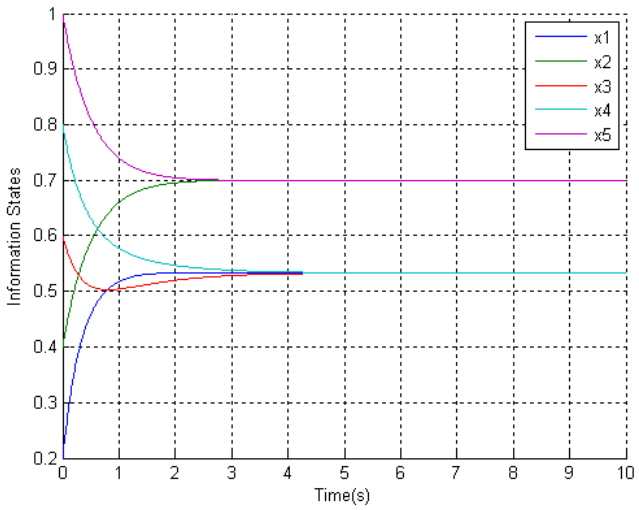

Figure 2.10. Average consensus for a graph with two strongly connected components.

2.6.2. Future Research of Connectivity Analysis of Consensus Algorithms

So far, the connectivity analysis of consensus algorithms is examined for directed graphs which have strongly connected components. In particular, algorithm given in Equation 2.10 is analyzed under fixed interaction topology given in Figure 2.9 and a convergence result related to the consequence given in Lemma 2.3 is obtained.

The results of the present work point to a promising research direction at the intersection of graph theory and control theory that aims to study connectivity attributes of consensus algorithms. Moreover, future research involves extending the proposed framework not only to strongly connected and balanced graphs but also to unbalanced graphs, directed graphs which have spanning trees and the graphs concern with multiple consensus equilibriums.

2.7. Summary of the Chapter

In this chapter, after giving an overview of Graph Theory, fundamentals of consensus algorithms are studied. Consensus for both fixed and dynamically changing interaction topologies are examined with the help of convergence analysis. Each case is illustrated by numerical examples. In consequence, average consensus is achieved if and only if the directed communication topology is both strongly connected and balanced. In contrast, when the directed communication topology contains a rooted directed spanning tree, the consensus equilibrium is equal to the weighted average of the initial states which is a less stringent condition. In the last section of the chapter, connectivity analysis of consensus algorithms with numerical examples is described.

3. GRAPH-BASED FORMATION CONTROL FOR MOBILE ROBOT NETWORKS

Communication capabilities of the agents and sensing ranges are some of the necessary data for designing control strategies for mobile robot networks. However, this can be incorporated at a lower level and abstracted to a simple definition of which links between agents exist over time. This type of approach is known as the graph-based approach to multi-agent control design and is prevalent in many control applications.

Modeling the system with a graph-based framework where communication links are captured in the combinatorial objects associated with a graph such as the edge and vertex sets is most of the time advantageous. In order to capture the network topology of a multiple mobile robot system, graph models including static graph that the communication links are stable over time are commonly used. Despite its simplicity, this model often does not reflect reality. With the aim of having a more realistic model, a model including a random graph in which communication links are modeled using stochastic processes is often used. Additionally, in a proximity graph, the existence of a communication link depends on a constraint that the distance between the two agents must satisfy [6]. Once the graph model is derived, it becomes possible to apply well-established algebraic graph theory on it, so that there will be meaningful conclusions about the robotic system [27, 28].

Formation control of a mobile robot network has been an active topic of recent research. To analyze the stability properties of formations, Fax *et al.* related the eigenvalues of the formation graph Laplacian to formation stability in [9]. Furthermore, a Nyquist plot, which can be used to quantify stability, can be arisen from the formation graph and eigenvalues of the Laplacian in [7].

Previously mentioned works have shown how to use the undirected and directed graphs to represent multi-agents networks, their topology and communication. It

has also been studied how to control formations and achieve consensus using graph-theoretic models. The study in [5] takes the research one step further with the introduction of a moving leader into 2D formations of holonomic agents. The paper is mainly based on the notable results that have a well described summary in [7] by Olfati-Saber *et al.* and the graph-based models to control a desired formation, representing the interactions and the flow of information between the multiple agents in the graph is explored by the authors. The solution presented is to add a learning term that gives a first order approximation of the general movement, that is, a velocity estimation of the leader. In particular, the main aim is to achieve and maintain a formation from any initial condition, with and without a leader that the entire formation must follow.

Another topic that is investigated is hybrid control strategies in which the robots switch between different formations. In [29], the authors are interested in achieving a formation while ensuring that the agents stay in the convex polytope for all times, which is called containment control, spanned by dedicated leader-agents, whose dynamics will be given by a hybrid Stop-Go policy. In [30], a strategy for switching formation modes is presented by Axelsson et al. This strategy is based on introducing a formation error associated with each possible formation. The robots autonomously decide to execute the formation which results in the smallest error. Each robot uses a hybrid automation to decide which formation to execute, and state transitions are triggered by minimizing a cost function associated with formation error.

Recently, some papers are focused on the issue of preserving connectedness through nonlinear weights. The results of different works can be sorted by if either connectedness is assumed [31,32], or connectedness is only required at distinct times [12,13,19,33] in the sense that the agents sense their environment and then move in such a way that the network is connected at the sensing times, where the agents may be operating synchronously or asynchronously. More particularly, Ando *et al.* gave the first solution to the connectedness preserving rendezvous problem in [33], this solution is a discrete-time control algorithm is proposed that evaluated and ensured connectivity, as well as other constraints, at each instant of discrete time. Another research with relevant contribution thanks to its results is described in [34]. This one deals with

the connectivity-maintenance problem for ad hoc networks with discrete-time double-integrator dynamics. However, in [27], M. Ji and M. Egerstedt showed how to make the graph stay connected for all times by studying a collection of graph-based nonlinear feedback control laws for distributed multi-agent systems where the nonlinear feedback laws are based on weighted graph Laplacians and they are proved to be able to solve the rendezvous and formation-control problems while ensuring connectedness.

Another novel topic is the issue of how the ratio of leaders-to-followers has an impact on connectivity while many issues regarding controllability and stability of leader-follower networks. Couzin et al. described some extended numerical results on this issue in [35]. But in [17], the problem is treated from an analytical standpoint. Moreover, the situation is considered in [27, 34] where connectivity is maintained in networks of homogenous agents directly through the control law. On the contrary, in [17], indirect metrics to guarantee connectivity maintenance and rendezvous in a leader-follower network are provided. These metrics are the leader to follower ratio and the parameters of the goal attraction function. It is well known that a general connectivity analysis of an arbitrary network is extremely complex, but in [17], the authors showed how network structure can be used to obtain interesting results for some special classes of networks for the rendezvous objective.

3.1. Graph-Based Formation Control

The block diagram for a network of interconnected dynamic systems all with identical transfer functions $P(s) = 1/s$ is shown in Figure 3.1. The collective networked system has a diagonal transfer function and is a MIMO (multi-input multi-output) linear system. An iterative form of the consensus algorithm given in 2.10 can be stated as follows in discrete-time:

$$x_i(k+1) = x_i(k) + \epsilon \sum_{j=1}^{n} a_{ij}(x_j(k) - x_i(k)) \tag{3.1}$$

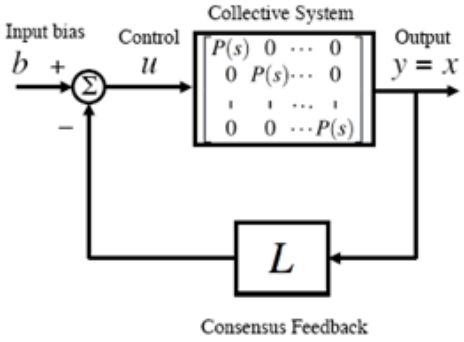

Figure 3.1. Block diagram of the collective networked system (reprinted from [7]).

The discrete time collective dynamics of the network under this algorithm can be written as

$$x(k+1) = Px(k) \tag{3.2}$$

with $P = I - \epsilon L$ where I is the identity matrix and the step-size $\epsilon > 0$. Differently from the problem addressed before, in order to obtain a formation, a bias is added to discrete-time consensus algorithm:

$$x_i(k+1) = x_i(k) + \epsilon \sum_{j=1}^{n} a_{ij}(x_j(k) - x_i(k) - r_{ij}) \tag{3.3}$$

The value r_{ij} represents the desired inter-agent relative position between i and the neighbor j. The collective dynamics can be written as :

$$x(k+1) = Px(k) + \epsilon diag(AR) \tag{3.4}$$

The matrix R is composed by the elements r_{ij} that contain information about the formation and A is the adjacency matrix that constrains the information flow between agents.

3.1.1. Leader Based Multi-Agent Control

When a leader is inserted into the network, the digraph is no longer strongly connected, but the consensus algorithm remains valid in the sense that, when the leader stops, the formation will be reached, and until then, the followers will always try to form up, using their current relative position to the leader and among each other to correct their locations.

In this section, the problem is addressed with n + 1 agents. It is considered that there is an agent called leader whose movement is not determined by the consensus equation but instead by an external input or a predefined velocity and/or path pattern. The remaining n agents, named followers are ruled by the consensus equation and must follow the leader while preserving a formation. Therefore, with the introduction of the leader we gain the possibility to move the formation [5].

The solution proposed for this new problem is that every agent in the network will learn the average neighbor velocity, which will converge to the average leader velocity. Arbitrating that the leader is node n+1, this learning procedure is implemented, through the inclusion of a new term in discrete-time consensus algorithm leading to

$$x_i(k+1) = x_i(k) + \epsilon \sum_{j=1}^{n} a_{ij}(x_j(k) - x_i(k) - r_{ij}) + \xi_i(k+1), i = 1, 2, ..., n \tag{3.5}$$

$$x_{n+1}(k+1) = x_{n+1}(k) + v_{leader} \tag{3.6}$$

where N_i is evaluated for all the n+1 agents. The collective dynamics of the followers can be written as

$$x_F(k+1) = P x_F(k) + \epsilon\, diag(AR) + \xi(k+1) \tag{3.7}$$

where $x_F(k)$ are the first n elements of $x(k) \in \mathbb{R}^{n+1}$ and the matrices A, R and P are

the same as above. $\xi \in R^n$ encodes the average velocity of the neighbors.

3.2. Convergence Analysis

In this section, we investigate the convergence properties of the consensus algorithm stated in Equation 3.1 in discrete time. The corresponding discrete-time collective dynamics is also stated in Equation 3.2. Before presenting the framework for analysis of convergence of consensus algorithms, we first give detailed information on the matrix P which is called as Perron matrix of a graph G with parameter $\epsilon > 0$ representing the step-size.

There are three important special cases of nonnegative matrices which are irreducible, stochastic, and primitive or ergodic matrices. A matrix A is said to be irreducible if its corresponding graph representation is strongly connected. A nonnegative matrix is said to be row (or column) stochastic when all of its row-sums (or column-sums) are equal to 1. An irreducible stochastic matrix P is called primitive or ergodic if it has only one eigenvalue with maximum modulus [18].

Lemma 3.1. *Let G be a digraph with n nodes and maximum degree $\Delta = max_i(\sum_j a_{ij})$. Then, the Perron matrix P with parameter $\epsilon \in (0, \frac{2}{\Delta}]$ satisfies the following properties [7]:*

- *P is a row stochastic nonnegative matrix with a trivial eigenvalue of 1,*
- *If G is a balanced graph, then P is a doubly stochastic matrix,*
- *If G is strongly connected and $0 < \epsilon < \frac{2}{\Delta}$, then P is a primitive matrix.*

Applying Gershgorin circle theorem stated in Theorem 2.2 to the Perron matrix, we determine that the eigenvalues are in the following region

$$\bigcup_{i=1}^{n} \{z \in C : |z - p_{ii}| \leq 1 - p_{ii}\} \tag{3.8}$$

where p_{ii} is the ith diagonal element of P.

It is obvious from Figure 3.2 that the union of these regions is the region surrounded by the circle with the largest radius. Since the radius is defined by $1 - p_{ii}$, the smaller diagonal element of the Perron matrix P defines the region where all eigenvalues are inside. Proofs of these properties are given in [7].

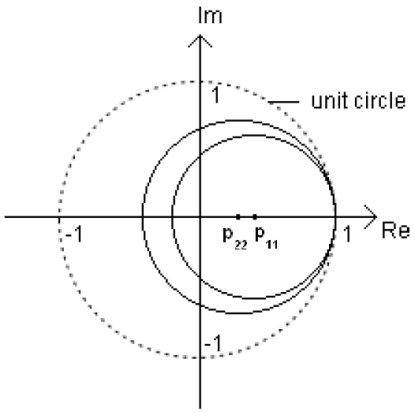

Figure 3.2. Gershgorin circles of the Perron matrices.

The condition $\epsilon < \frac{2}{\Delta}$ given in the last property is necessary. If an incorrect step-size of $\epsilon = \frac{2}{\Delta}$ is used, then, P would no longer be a primitive matrix because it could have multiple eigenvalues of modulus 1. Lemma 3.2 and Theorem 3.1 decribe the convergence analysis of the discrete-time consensus algorithm.

Lemma 3.2. *Let P be a primitive nonnegative matrix with left and right eigenvectors w and v, respectively, satisfying $Pv = v$, $w^T P = w^T$, and $v^T w = 1$. Then $lim_{k\to\infty} P^k = vw^T$, [18].*

Hence the following result states the convergence and group decision properties of iterative consensus algorithms with row stochastic Perron matrices. The statement and proofs of these convergence analysis results are given in [7].

Theorem 3.1. *Consider a network of agents $x_i(k+1) = x_i(k) + u_i(k)$ with topology*

G applying the distributed consensus algorithm

$$x_i(k+1) = x_i(k) + \epsilon \sum_{j \in N_i} a_{ij}(x_j(k) - x_i(k)) \tag{3.9}$$

where $0 < \epsilon < \frac{2}{\Delta}$ *and* Δ *is the maximum degree of the network. Let* G *be a strongly connected directed graph. Then*

- *A consensus is asymptotically reached for all initial states,*
- *The group decision value is* $\alpha = \sum_i w_i x_i(0)$ *with* $\sum_i w_i = 1$,
- *If the directed graph is balanced or in other words* P *is doubly-stochastic, then an average consensus is asymptotically reached and* $\alpha = \frac{\sum_i x_i(0)}{n}$.

3.3. Convergence Speed Analysis

In a network, the total time required to achieve a consensus is called *convergence time*. The speed of reaching consensus is the key in design of the network topology as well as analysis of performance of a consensus algorithm for a given network. Therefore faster convergence is always a desired performance metric for consensus algorithms. In this section we analyse the relationship between the structure of the Perron matrices and the convergence speed, and illustrating numerical examples will be given.

In Section 3.2, it is proven that the largest eigenvalue of a Perron matrix is one. Moreover, all other eigenvalues are less than one. Let λ_2 be the second largest eigenvalue, i.e.

$$\lambda_1 = 1 > |\lambda_2| \geq ... \geq |\lambda_n| \tag{3.10}$$

Then the convergence speed is related to λ_2 and a smaller λ_2 provides faster convergence. The proofs are stated in [8].

3.3.1. Numerical Examples for Performance Analysis of Consensus Algorithms

Example 3.1. In this example, L is fixed in order to see the effect of the parameter ϵ. Consider the Laplacian matrix as follows

$$L = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

$\epsilon_1 = 0.01$ and $\epsilon_2 = 0.49$ are chosen between the interval $[0; \frac{2}{\Delta}]$. All eigenvalues of the Perron matrix are in $[1 - \epsilon\lambda_{max}(L); 1]$. The second largest eigenvalues $\lambda_2(P_1)$ and $\lambda_2(P_2)$ are respectively 0.99 and 0.51. Hence, the more we increase the value of ϵ, the faster we get to the convergence. As expected, since $\lambda_2(P_2) < \lambda_2(P_1)$, the second system converges faster.

Example 3.2. In this example, ϵ is fixed to 0.1 and L is normalized by the outdegree.

$$\bar{L} = \begin{bmatrix} 1 & -1/2 & -1/2 & 0 \\ -1/2 & 1 & -1/2 & 0 \\ -1/3 & -1/3 & 1 & -1/3 \\ 0 & 0 & -1 & 1 \end{bmatrix}, P = \begin{bmatrix} 9/10 & 1/20 & 1/20 & 0 \\ 1/20 & 9/10 & 1/20 & 0 \\ 1/30 & 1/30 & 9/10 & 1/30 \\ 0 & 0 & 1/10 & 9/10 \end{bmatrix}$$

The second largest eigenvalue of P corresponding to L and $\bar{L}$ are 0.9 and 0.9229 respectively. Since the latter second largest eigenvalue is larger than the former, the convergence speed of the system with $\bar{L}$ is slower. Moreover, since the column-sum of the P is not equal to 1 the system is no longer balanced. As a result, the final state of the consensus corresponding to the normalized case is changed.

Example 3.3. In this example, we consider the Laplacian matrix $\bar{L}$ and chose ϵ to be larger than $\frac{2}{\Delta}$, 0.99, so that the diagonal values of P are close to 0.

$$P = \begin{bmatrix} 1/100 & 99/200 & 99/200 & 0 \\ 99/200 & 1/100 & 99/200 & 0 \\ 33/100 & 33/100 & 1/100 & 33/100 \\ 0 & 0 & 99/100 & 1/100 \end{bmatrix}$$

The second largest eigenvalue of P is 0.2364. This value is the smallest that we got among all numerical examples. Hence, the convergence speed of this numerical example is the fastest. Like in the previous example, the column-sum of P is not equal to 1. As a result, the final state of the consensus is changed.

From the above examples, we note that choosing the averaging weights i.e. the components of the P matrix, has a drastic effect on the convergence speed of the formation algorithm. In [36, 37], this problem is investigated in some detail. In [37] it is shown that this is indeed an open problem even in centralized case.

3.4. Formation Control Simulations and Results

In this section simulation results that illustrate the various aspects addressed in the previous section are presented and discussed by evaluating the performance of the consensus equations on two types of systems: a point-mass holonomic robotic agents and dynamically simulated omni-wheeled holonomic robotic agents.

3.4.1. Point Mass Simulations of Holonomic Robotic Agents

In Section 3.4.1.1., the results presented illustrate the behaviour of the agents when trying to reach consensus without a leader. In Section 3.4.1.2., the performance of the improved consensus equations proposed in Equation 3.5, Equation 3.6 and Equation 3.7 is presented in different motions of leader.

3.4.1.1. Distributed Consensus Without a Leader Case. In order to present the results of the simulations, it is necessary to state the network configurations used in

the cases. The initial positions of the agents are $x_1 = (-2, 2)$, $x_2 = (-1.5, -1.5)$, $x_3 = (1.5, 0.5)$, $x_4 = (2, -2)$. Moreover, the possible flows of communication between the nodes are represented by arrows, expressing the edges of the graph in Figure 3.3.

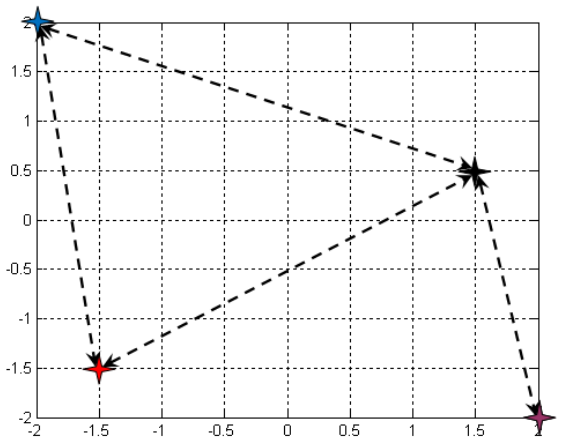

Figure 3.3. Graph representation of the network placed in the initial positions.

In Figure 3.4, the simulated results are represented using the graph presented in Figure 3.3. The distributed consensus algorithm used to drive the agents to the common final location, at (0,-0.25), the average consensus of the agents initial position.

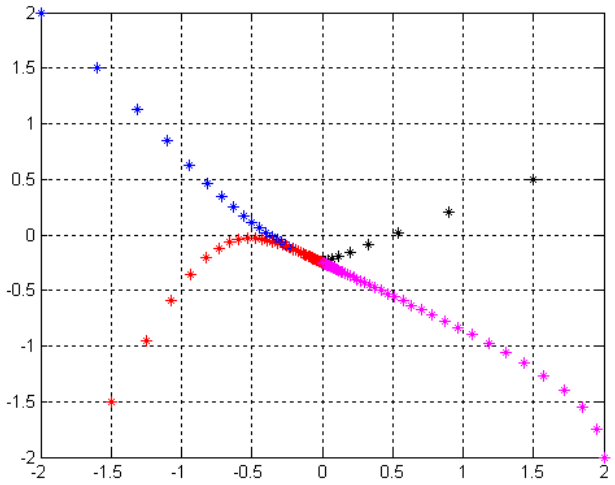

Figure 3.4. Agents reaching an average consensus.

Figure 3.5 shows the geometry of a desired formation to the network. This topology is achieved using the elements r_{ij}:

$$R_x = \begin{bmatrix} 0 & -0.5 & 0.5 & 0 \\ 0.5 & 0 & 1 & 0.5 \\ -0.5 & -1 & 0 & -0.5 \\ 0 & -0.5 & 0.5 & 0 \end{bmatrix}, R_z = \begin{bmatrix} 0 & -0.75 & -0.75 & -1.5 \\ 0.75 & 0 & 0 & -0.75 \\ 0.75 & 0 & 0 & -0.75 \\ 1.5 & 0.75 & 0.75 & 0 \end{bmatrix}$$

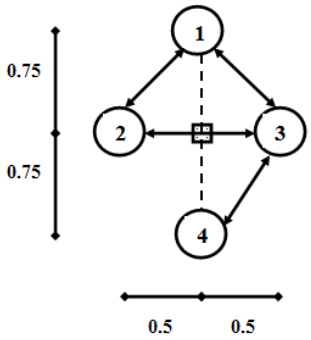

Figure 3.5. Target relative positions of the rhombus formation.

Figure 3.6 illustrates the time behavior of the system until it reaches the final formation in Figure 3.5, using the consensus algorithm.

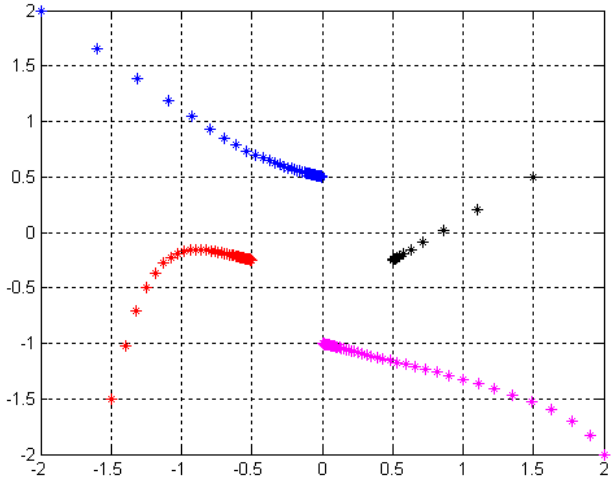

Figure 3.6. Agents reaching a formation.

3.4.1.2. Leader-Based Distributed Formation. The main goal in this section is to present results of the consensus algorithm with different motions of the leader. The final for-

mation that the network has to reach and maintain while following the leader is the one presented in Figure 3.5 as node 1.

In Figure 3.7, the leader is moving with constant velocity along the $y = x$ line and the final position is shown. The agents in the network reach the desired formation and follow the leader.

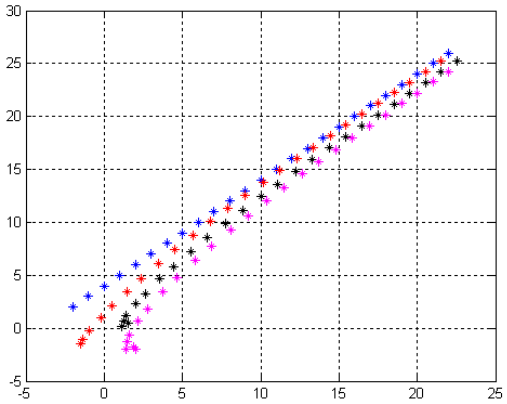

Figure 3.7. Linear trajectory.

Figure 3.8 presents the network's behavior when the leader is performing a counter clockwise circular trajectory, again with constant velocity. The followers attempt to reach the formation and reproduce the same motion of the leader, with good results.

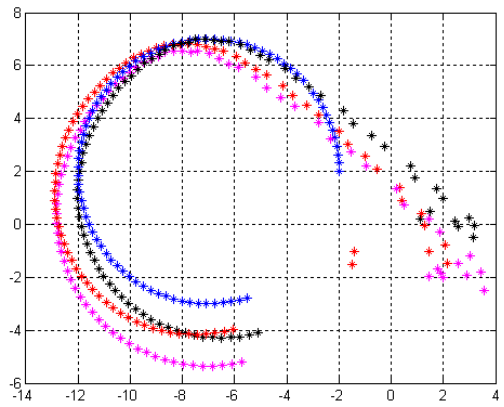

Figure 3.8. Circular trajectory.

3.4.2. Dynamic Simulation of Omni-Wheeled Holonomic Robotic Agents

The solution of the graph-based distributed formation control problem briefly outlined in Section 3.4.1 is based on the assumption that the agents are modelled as point-mass holonomic agents. When dealing with real robots, this assumption does not hold, since in reality robots have some dynamics, e.g., the robots have limited acceleration and velocity because of the environmental conditions such as friction. Furthermore, each robot has different center of mass and moment of inertia due to the mechanical constraints. A typical example of a real robot modelled for the RoboCup SSL Competition by Brocks in 2010 is depicted in Figure 3.9. Notably, in contrast

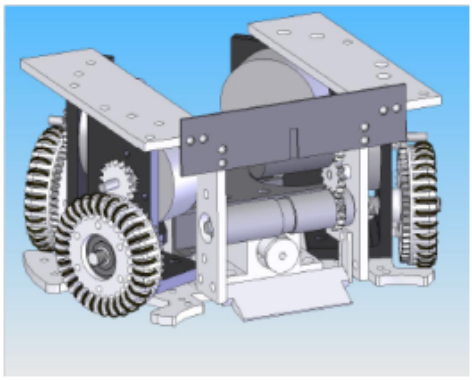

Figure 3.9. Technical drawing of Four Wheeled Robot Wheelbase [40].

to previous simulations, the consensus equations are used to control a distributed formation of multi-agent system where the agents have significant dynamics. Due to this, the following section starts by presenting the modelling of a four wheeled omni-directional robot.

3.4.2.1. Modelling of a Four Wheeled Omnidirectional Robot. Omnidirectional robots have become popular in the small size league of the annual RoboCup competition since they are very maneuverable in a highly dynamic environment. Particularly, omnidirectional wheels allow the robot to navigate on a given straight path without having to rotate.

The general working principle of omni-directional locomotion system is based

on the frictionless slide in the motor axis direction. In order to achieve this, each of the wheels is composed of smaller wheels attached along the periphery. Figure 3.12 illustrates an example of omnidirectional wheel.



Figure 3.10. Locomotion System: Omnidirectional wheel [40].

The dynamical equations relative to the translational accelerations a_x, a_y and the angular acceleration α of the center of mass of the robot are given in the following expression which is detailed in [41]. Figure 3.11 shows the related geometry of omnidirectional robot.

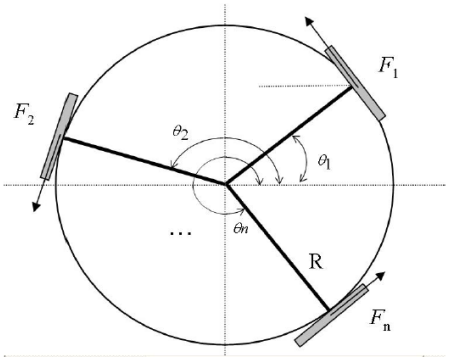

Figure 3.11. Arrangement of n omnidirectional wheels [41].

$$\begin{pmatrix} a_x \\ a_y \\ \alpha \end{pmatrix} = \frac{1}{M} \begin{pmatrix} -sin\theta_1 & -sin\theta_2 & ... & -sin\theta_n \\ -cos\theta_1 & -cos\theta_2 & ... & -cos\theta_n \\ \frac{MR}{I} & \frac{MR}{I} & ... & \frac{MR}{I} \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} \tag{3.11}$$

where M is the mass, R is the radius of the robot, I is the moment of inertia, and f_i's denote the magnitude of the force vectors F_i.

Dealing with the kinematic model of an omnidirectional robot, the final velocities of the wheels, and the velocity of the robot on the plane, as well as its angular velocity, are computed by integrating the dynamical equations given above with respect to time. However, it should be noted that the robots trajectory have to be computed in Euclidean space where the velocity of each individual wheel is derived.

The following expression is examined in detail in [41] for the correspondences between the Euclidean and motor speeds. The equations are given as follows where the individual speeds of the n motors is the vector $(v_1, v_2, ..., v_n)^T$ and the Euclidean velocity and tangential rotational speed of the robot is the vector $(v_x, v_y, R\omega)^T$.

$$\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} -sin\theta_1 & cos\theta_1 & 1 \\ -sin\theta_2 & cos\theta_2 & 1 \\ \vdots & \vdots & \vdots \\ -sin\theta_n & cos\theta_n & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ R\omega \end{pmatrix} \tag{3.12}$$

The positive rotation direction is the direction of the right hand thumb when we hold the motor axis in the hand. According to this convention, the following expression for the correspondences between the Euclidean and motor speed is obtained for four wheeled omnidirectional symmetric robot where in our study φ is equal to $\frac{\pi}{4}$:

$$\begin{pmatrix} v_1, & v_2, & v_3, & v_4 \end{pmatrix}^T = \begin{pmatrix} -sin\varphi & cos\varphi & 1 \\ -sin\varphi & -cos\varphi & 1 \\ sin\varphi & -cos\varphi & 1 \\ sin\varphi & cos\varphi & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ R\omega \end{pmatrix} \tag{3.13}$$

To understand the effects of the underlying dynamics, the performances of the

consensus equations on the dynamically simulated robots are examined in Webots which has become a popular platform for simulating cooperative problems such as formation control [42], navigation [43] and task allocation [44] in recent years.

3.4.2.2. Webots: A Simulation Environment. Webots is a professional mobile robotics simulation package distributed by Cyberbotics [45]. It provides 3D visualisation of simulation environments and utilises the Open Dynamics Engine (ODE) for rigid body dynamics and collision detection. It actually allows the designers to visualize rapidly their ideas, to check whether they meet the requirements of the application, to develop the intelligent control of the robots, and eventually, to transfer the simulation results into a real robot.

Webots allows the user to perform 4 basic stages in the development of a robotic project as depicted in Figure 3.12.

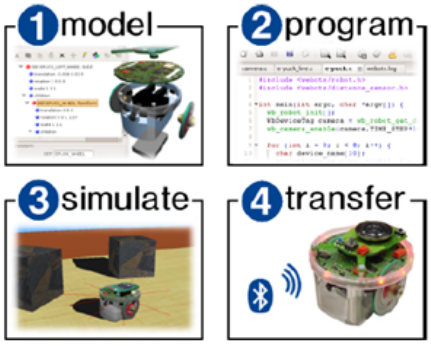

Figure 3.12. Webots development stages [45].

The first stage is the modelling stage which consists in designing the physical body of the robots, including their sensors and actuators and also the physical model of the environment of the robots. Programming stage is the second stage that the behaviour of each robot has to be programmed from a C, C++ or Java file. The third stage is the simulation stage which allows the user to test if the program behaves correctly. Finally, the fourth stage is the transfer to a real robot that running in the real world.

In Webots, physics can be extended with a custom plug-in to allow simulation of environments such as air or water. Moreover, custom robots can be created using a library of geometries, sensors and actuators, or one of the supplied robot models can be used. Robot position information is provided to robots in this thesis using the supervisor which is a special robot class that provides functions such as changing the environment, sending messages to robots or recording trajectories.

In this study, we designed the body of the agents as a four wheeled, omnidirectional and symmetric shape. In order to analyze the algorithms clearly, all agents are designed as identical to each other. The value of acceleration parameter of each wheels' servo nodes is taken as -1 which means that the acceleration is infinite. By this way, force of friction is neglected so that only kinematic equations are considered. The design parameters are given in Figure 3.13.

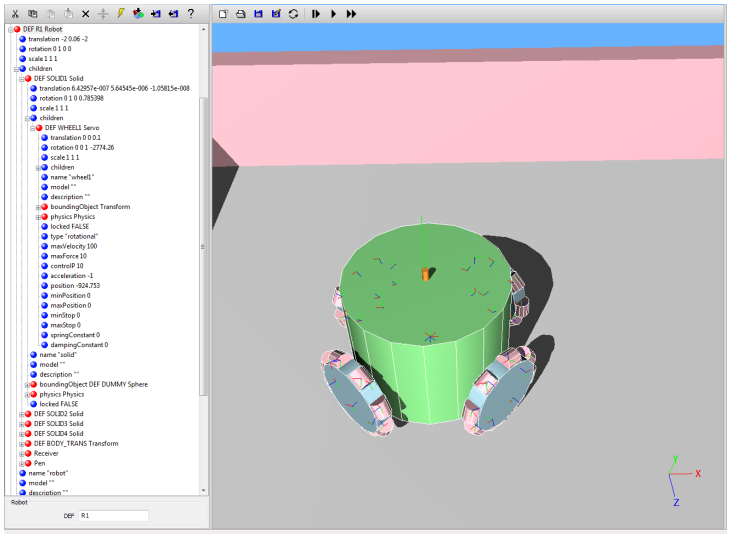

Figure 3.13. Webots environment of the study.

3.4.2.3. Distributed Consensus Without a Leader Case. Firstly, the initial positions of the agents are chosen as the same as in section 3.4.1: $x_1 = (-2, 2)$, $x_2 = (-1.5, -1.5)$, $x_3 = (1.5, 0.5)$, $x_4 = (2, -2)$. The network flows of communication between the nodes are shown in Figure 3.3. The distributed consensus algorithm used to drive the agents

to the common final location at $(0, -0.25)$, the mean point of the agents initial positions. In Figure 3.14, the related Matlab and Webots simulation results are represented respectively.

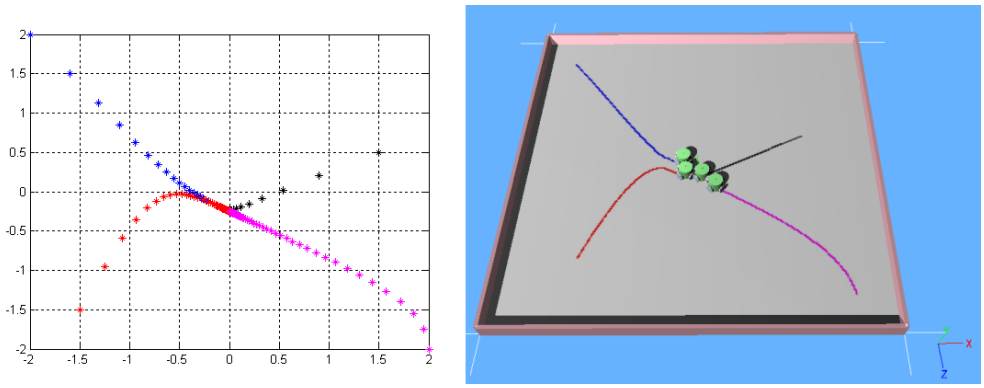

Figure 3.14. Consensus Simulation in (a) Matlab (b) Webots.

Secondly, target relative positions of the formation and the network flows of communication between the nodes are shown in Figure 3.5. The time behavior of the system that reaches the final formation using the consensus algorithm is illustrated in Figure 3.15 that contains Matlab and Webots simulation results.

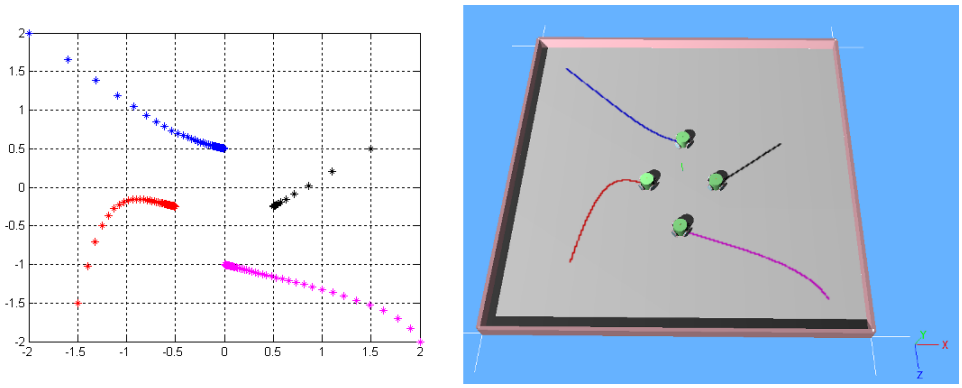

Figure 3.15. Formation Simulation in (a) Matlab (b) Webots.

3.4.2.4. Leader Based Distributed Formation Case. Finally the time behavior of the system when the leader is following the line $y = x$ with constant velocity is illustrated in Figure 3.16 that contains Matlab and Webots simulation results. The agents in the network reach the desired formation and follow the leader.

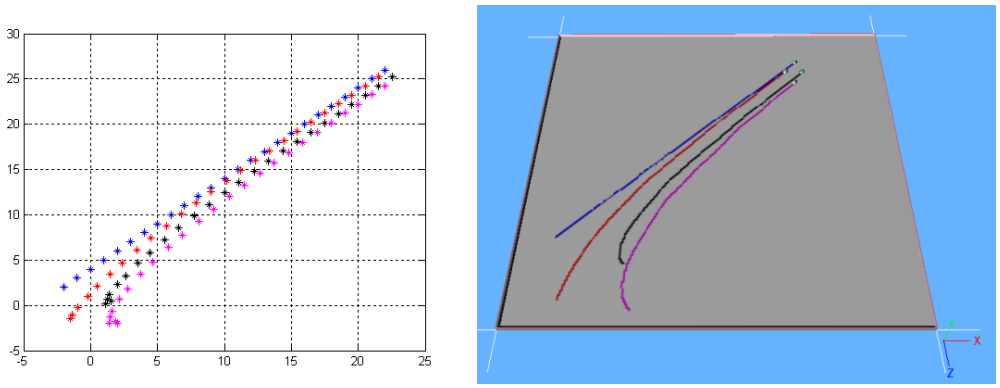

Figure 3.16. Linear Trajectory Simulation in (a) Matlab (b) Webots.

3.5. Summary of the Chapter

In this chapter, first graph-based distributed formation control is explained. Then performance analysis of consensus algorithm for three different cases illustrated with numerical examples. Furthermore, formation control simulations and results have been given for point mass holonomic robotic agents and omni-wheeled holonomic robotic agents with and without a leader. For dynamic simulation of omni-wheeled holonomic robotic agents, Webots Environment has been used. Building on the results of this chapter, a combination of leader-based distributed formation control with a type of motion planning will be explained in the next chapter.

4. MOTION PLANNING

Autonomous robotics has some difficulties, most notably in the area of automatic motion planning, where the objective is to identify a task in a high-level language and make the robot automatically organize this specification into a set of low-level motion primitives, or feedback controllers, to achieve the task. The given task is to find a path for a robot, whether it is a robot arm, a mobile robot, from one configuration to another while keeping itself away from obstacles.

In this chapter we will analyze motion planning algorithms for the leader based distributed formation control. In the previous chapter we only focused on the algorithms for the leader and the followers that move with constant velocity. However, motion planning provides a path that enables leader as well as followers avoid obstacles in the environment while maintaining formation. Before the combination of path planning and distributed formation control is examined, we are going to provide detailed information about concepts in motion planning.

4.1. Overview of Concepts in Motion Planning

In this section, some of the important concepts are briefly explained. The motion planner is grouped into three categories such as the task it addresses, properties of the robot solving the task, and properties of the algorithm.

The most important category related with a motion planner is the task it addresses, i.e. the problem it solves. There are mainly four different tasks: *navigation*, *coverage*, *localization*, and *mapping*. *Navigation* is the problem about finding a collision-free motion for the robot system from one state to another. The robot can be for example thought as a robot arm, or a mobile robot. *Coverage* is the problem about passing a sensor or tool over all points in a space, such as in demining or painting. *Localization* is the problem related with using a map to interpret sensor data to determine the configuration of the robot. *Mapping* is the problem about exploring

and sensing an unknown environment to construct a representation that is useful for navigation, coverage, or localization.

4.1.1. Properties of the Robot

The efficiency of motion planner depends on properties of the robot solving the task. For instance, the robot and the environment determine the number of *degrees of freedom* of the system and the shape of the *configuration space*. Firstly, the robot's configuration space should be studied and understood in detail. It could be analysed according to the fact that the robot is free to move instantaneously in any direction in its configuration space without any obstacles. This is called an omnidirectional robot. If the robot is exposed to velocity constraints, such as a car that cannot translate sideways, both the constraint and the robot are called *nonholonomic*. Finally, we can model the robot by using *kinematic* equations, with velocities as controls, or by using *dynamic* equations of motion, with forces as controls.

4.1.2. Properties of the Algorithm

Firstly, the task and the robot system are identified, then the algorithm can be chosen according to the criteria which depends on how they solve the problem. For instance, questions about *optimality* in length, execution time, or energy consumption can be considered. In addition to the quality, some questions, which are related with the *computational complexity* of the planner, can be asked in order to make an effective analysis. For example, the planner can wonder if the memory requirements and running time of the algorithm are constant, polynomial, or exponential in the "size" of the problem description. The size of the input could be the number of degrees of freedom of the robot system, the amount of memory needed to describe the robot and the obstacles in the environment, etc., and the complexity can be defined according to the worst case or the average case. Some planners are said to be *complete* which means that a solution related with the motion planning problem when one exists or indicate failure in finite time can be always found. This is a very powerful and desirable property. When the number of degrees of freedom increases, complete solutions may

become computationally intractable.

4.2. Configuration Space

In order to create motion plans for robots, position of the robot has to be specified. More specifically, the specification of the location of every point on the robot has to be given, due to the fact that is has to be ensured that no point on the robot collides with an obstacle.

In this section, configuration of a robot is discussed and the concept of the configuration space is introduced. Configuration concept is one of the key concepts in robot motion planning. The *configuration* of a robot system is a complete specification of the position of every point of that system. The *configuration space*, or *C-space*, of the robot system is the space of all possible configurations of the system. Thus a configuration is simply a point in this abstract configuration space. Throughout this section, q denotes a configuration and to denote the configuration space. The number of *degrees of freedom* of a robot system is the dimension of the configuration space, or the minimum number of parameters which are used to specify the configuration.

To solve path planning problems, there are on one side classical approaches including potential field path planning, roadmaps and cell decomposition and on the other side sampling-based algorithms including Probabilistic Road map Method (PRM) and Rapidly Exploring Dense Trees (RDT).

4.3. Potential Field Path Planning

Potential field path planning consists of generating a field, also called gradient, across the robots map which will lead the robot from different prior positions towards the goal position [46]. This method was originally created for robot manipulator path planning. In this method, the robot is considered as a point under the impact of an artificial potential field denoted by $U(q)$ (Figure 4.1). Just like a ball rolling downhill, the robot moves by following the field. In this space, the goal as a minimum acts like an

attractive force in contrast to obstacles acting like repulsive forces, also called peaks, on the robot. The robot, most commonly represented as a point, is under the impact of the superposition of all forces applied. In this way, the robot is guided towards the goal while avoiding known obstacles at the same time.

It should be noted that this method is not only just path planning as the field generated acts also as a control law for the robot. In other words, with the assumption that the robot can localize its current position according to the map ant the potential field, its next required action can always be determined. In all potential field approaches, the robot is attracted towards the goal and at the same time repulsed by the obstacles known in advance. If, during robot motion, new obstacles come out one could always update the field to input the new information. In the simplest case, the robot is modeled as a point so it has no orientation and hence the potential field can be represented in 2D (x, y).

Let $U(q)$ be a differentiable potential field function, then the related artificial for $F(q)$ acting at position $q = (x, y)$ can be computed like:

$$F(q) = -\nabla U(q). \tag{4.1}$$

where $\nabla U(q)$ denotes the gradient vector of U at position q.

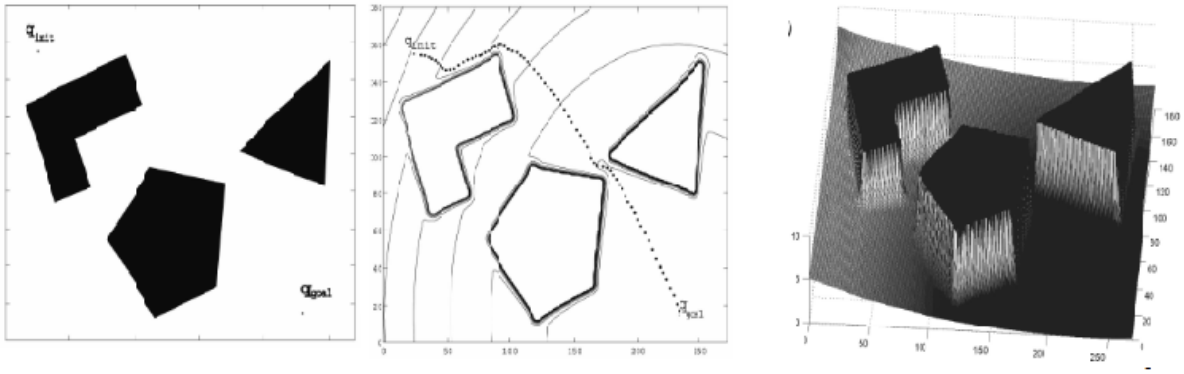

Figure 4.1. Typical potential field generated by the attracting goal and three obstacles. (a) Configuration (b) Equipotential plot (c) Resulting potential field [46].

$$\nabla U = \begin{bmatrix} \frac{\partial U}{\partial x} \\ \frac{\partial U}{\partial y} \end{bmatrix} \tag{4.2}$$

Then, the potential field acting on the robot is calculated as the sum of all fields, i.e. attractive for the goal and repulsive for the obstacles:

$$U(q) = U_{att}(q) + U_{rep}(q) \tag{4.3}$$

By the same way, forces can also be divided into two parts, an attracting one and a repulsive one:

$$F(q) = F_{att}(q) + F_{rep}(q) \tag{4.4}$$

$$= -\nabla U_{att}(q) - \nabla U_{rep}(q) \tag{4.5}$$

4.3.1. Attractive Potential

One way of defining an attractive potential is to use a parabolic function as follows

$$U_{att}(q) = \frac{1}{2} k_{att} \rho^2_{goal}(q) \tag{4.6}$$

where k_{att} is a positive scaling factor and $\rho^2_{goal}(q)$ represents the Euclidean distance $||qq_{goal}||$. This attractive potential is differentiable, leading to the attractive force F_{att}.

$$F_{att}(q) = -\nabla U_{att}(q) \tag{4.7}$$

$$= -k_{att} \cdot \rho_{goal}(q) \nabla \rho_{goal}(q) \tag{4.8}$$

$$= -k_{att} \cdot (q - q_{goal}) \tag{4.9}$$

It can be seen that F_{att} converges linearly to zero as the robot reaches the goal.

4.3.2. Repulsive Potential

Repulsive potential consists of creating a force away from all known obstacles. When the robot is far from the objects, this repulsive potential should have no impact on its moves whereas when the robot is close to the object then the repulsive potential should be very strong. An example of such a repulsive field is given below:

$$U_{rep}(q) = \begin{cases} \frac{1}{2}k_{rep}(\frac{1}{\rho(q)} - \frac{1}{\rho_0})^2 & \text{if } \rho(q) \leq \rho_0 \\ 0 & \text{if } \rho(q) \geq \rho_0 \end{cases} \tag{4.10}$$

where k_{rep} is again a scaling factor, $\rho(q)$ is the minimal distance from q to the object and ρ_0 the distance of influence of the object. U_{rep} is positive of zero and as q gets closer to the object, the function converges to infinity. $\rho(q)$ is differentiable everywhere in the space when the object boundary is convex and piecewise differentiable which generates a repulsive force F_{rep} of the form:

$$F_{rep}(q) = -\nabla U_{rep}(q) \tag{4.11}$$

$$= \begin{cases} k_{rep}(\frac{1}{\rho(q)} - \frac{1}{\rho_0})\frac{1}{\rho^2(q)}\frac{q - q_{obstacle}}{\rho(q)} & \text{if } \rho(q) \leq \rho_0 \\ 0 & \text{if } \rho(q) \geq \rho_0 \end{cases} \tag{4.12}$$

As a result, the robot represented as a point is exposed to the resulting force $F(q) = F_{att}(q) + F_{rep}(q)$ which repulses the robot away from the robot and moves it towards the goal as shown in Figure 4.1. Given ideal conditions, setting the robots velocity vector proportional to the field force vector, the robot can be guided toward the goal just like a ball rolling around obstacles and down a hill. Unfortunately, there are some disadvantages of this approach. Firstly, there can be the problem of local minima depending on the obstacles shape and size. Secondly, several minimal distances $\rho(q)$ could exist when the objects are concave. This would result in an oscillation between the two closest points to the object and hence the robot could get stuck then completeness would not be satisfied. For more detailed analyses of potential field characteristics refer to [46].

4.4. Road Map Path Planning

By road map approaches the connectivity of the robots free space in a network of 1D curves or lines, is captured and it is called road maps. Firstly, a road map, used as a network of road (path) segments for robot motion planning, is constructed. By this way, path planning is reduced to connecting the initial and goal positions of the robot to the road network. Between these two positions, robot searches for a series of roads in order to reach its goal position.

The robots configuration space is decomposed according to obstacle geometry by road map. The challenge is that a set of roads is constructed and these constructed roads enable the robot to go anywhere in its free space. This means that the number of total roads is minimized. In general, completeness is maintained in such decompositions throughout the true degrees of freedom of the robot have been captured with appropriate fidelity.

There are two road map approaches. The first one is *Visibility Graph* in which roads come as close as possible to obstacles and resulting paths are minimum-length solutions and the second one is the *Voronoi diagram* in which roads stay as far away as possible from obstacles.

4.4.1. Visibility Graph

The visibility graph for a polygonal configuration space C consists of edges joining all pairs of vertices that can see each other (including both the initial and goal positions as vertices as well). The unobstructed straight lines (roads) joining those vertices are obviously the shortest distances between them. The task of the path planner is thus to find the shortest path from the initial position to the goal position along the roads defined by the visibility graph like represented in Figure 4.2.

The main advantages of Visibility Graph are its simplicity for implemenetation especially when obstacles are polygons and its optimality characteristic. However,

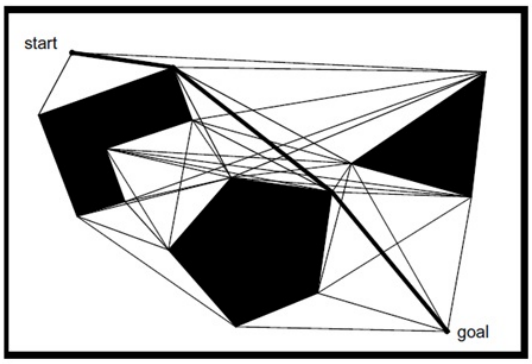

Figure 4.2. Visibility graph [46].

there are two important cautions when employing visibility graph search. The solution path found by the visibility graph tend to take the robot as close as possible to the obstacles, hence robots can collide. The common solution is to grow obstacles by more than robots radius. Furthermore it can be inefficient in densely populated environments

4.4.2. Voronoi Diagram

A Voronoi diagram is a complete road map method that maximizes the distance between the robot and obstacles in the map. With this specification, this diagram approach differentiates itself from the one of the visibility graph. For each point the distance to the nearest obstacle is computed in the free space. At points that have equal distance from two or more obstacles, the plot has sharp ridges. These sharp ridge points form the edges of the Voronoi diagram. When the configuration space obstacles are polygons, the Voronoi diagram consists of straight and parabolic segments.

The Voronoi diagram limited range localization sensors which is an important weakness of this approach. Since this path-planning algorithm maximizes the distance between the robot and objects in the environment, any short-range sensor on the robot will be in danger of failing to sense its surroundings.

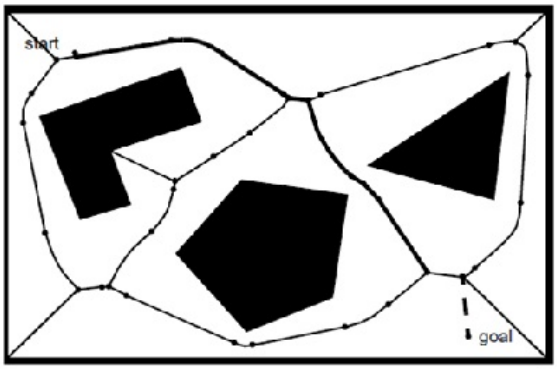

Figure 4.3. Voronoi diagram [46].

4.5. Cell Decomposition

Cell decomposition is used to discriminate between geometric areas, or cells, that are free or occupied by objects. The basic cell decomposition path-planning algorithm can be summarized as follows [46]:

- Divide into simple, connected regions called cells.
- Determine which opens cells are adjacent and construct a connectivity graph.
- Find the cells in which the initial and goal configurations lie and search for a path in the connectivity graph to join the initial and goal cell.
- From the sequence of cells which are found with an appropriate searching algorithm, compute a path within each cell, for instance, passing through the midpoints of the cell boundaries or by a sequence of wall-following motions and movements along straight lines.

The placement of the boundaries between cells is a crucial side of cell decomposition. If the boundaries are placed as a function of the structure of the environment, such that the decomposition is lossless, then the method is called exact cell decomposition. If the decomposition results in an approximation of the actual map, the system is called approximate cell decomposition.

4.5.1. Exact Cell Decomposition

Figure 4.4 reveals that the boundary of cells is based on geometric criticality. The basic generalization of a decomposition is that the particular position of the robot within each cell of free space is not important. Rather than that, the robots ability to traverse from each free cell to adjacent free cells is critical.

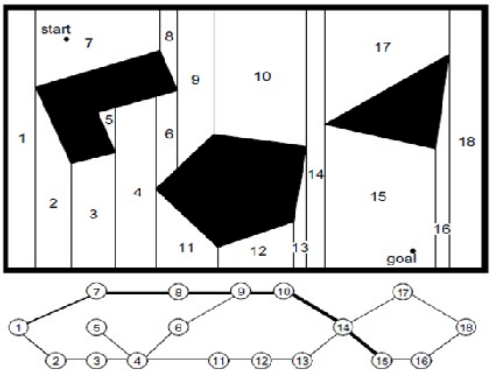

Figure 4.4. Exact cell decomposition example [49].

The important disadvantage of exact cell decomposition is the number of cells. Since overall path planning computational efficiency depends on the density and complexity of objects in the environment, the key crucial advantage is a result of this same correlation. In environments that are extremely sparse, the number of cells will be small, even if the geometric size of the environment is very large. Therefore the representation is efficient since the size of environments is large and sparse. The exact cell decomposition technique is used rarely in mobile robot applications due to complexities in implementation. However, it is a solid choice when a lossless representation is highly desirable, for example to preserve completeness fully.

4.5.2. Approximate Cell Decomposition

In mobile robot path planning studies, approximate cell decomposition approach is one of the most popular techniques owing to the popularity of grid-based environmental representations. These grid-based representations are fixed decompositions. Therefore, they are identical to an approximate cell decomposition of the environment.

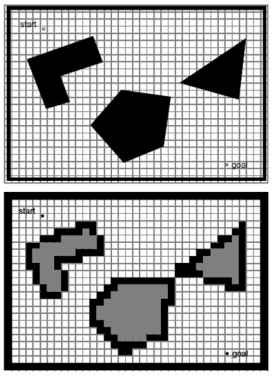

Figure 4.5. Grid based representation example [49].

Figure 4.5 reveals the most popular form of this approach, and it is called the fixed-size cell decomposition. The cell size does not depend on the particular objects in an environment, and therefore passageways can be lost due to the inexact nature of the pattern.

The crucial advantage of fixed size cell decomposition is the low computational complexity of path planning. NF1, or grassfire, is a good example of efficient and simple-to-implement algorithm for finding routes in such fixed-size cell arrays [48].

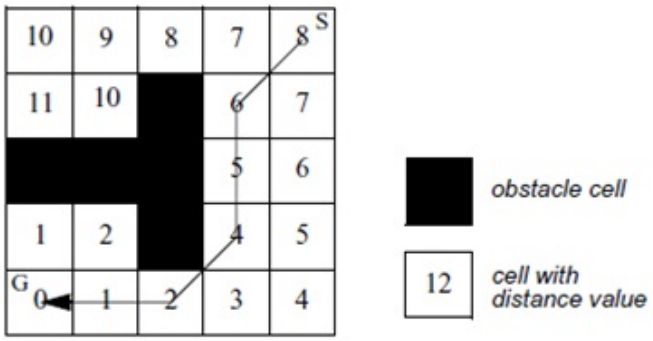

Figure 4.6. NF1 function example. S, start; G, goal [49].

When it is given that the entire array can be in memory, each cell is only visited once according to the shortest discrete path from the initial position to the goal position. Therefore, the search in the number of cells is linear. Unlike exact cell decomposition, this means that complexity does not depend on the sparseness and density of the environment. Furthermore, it does not depend on the complexity of the objects shapes in the environment. Formally, this grassfire transform is simply breadth-first search implemented in the constrained space of an adjacency array. Figure 4.7 reveals a variable-size approximate cell decomposition method. The free space is externally bounded by a rectangle and internally bounded by three polygons.

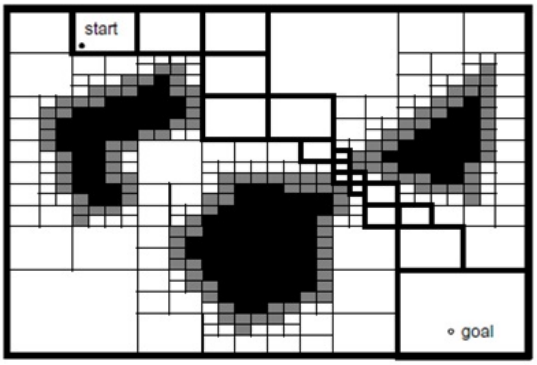

Figure 4.7. Approximate variable-cell decomposition example [49].

The rectangle which bounds the free space, is sliced into four identical rectangles. A rectangle is no longer decomposed when its interior lies completely in free space or in the configuration space obstacle. If the condition above is not satisfied, the decomposition process is repeated until some predefined resolution is attained. The white cells lie outside the obstacles, the black ones lie inside, and the gray ones are part of both regions.

When the exact cell decomposition method is compared with the approximate variable size approach, the latter one can sacrifice completeness, but it is easier to implement as it is mathematically less involving. In contrast to the fixed size cell decomposition, variable-size cell decomposition can be adapt to the complexity of the environment, and therefore sparse environments can contain appropriately fewer cells

and it can consume less memory.

4.6. Sampling-Based Algorithms

In earlier chapters, we described the different Planners in terms of their potential functions, cell decompositions and road maps in the free (or semi-free) configuration space. Each of these methods is based on an explicit representation of the geometry of C_{free}. Due to this property, when the dimension of the configuration space grows, these planners become impractical.

The main idea in sampling-based planning is to use the advances in collision detection algorithms that compute whether a single configuration is collision free. Given this simple primitive, a planner samples different configurations to construct a data structure that stores 1-D $C-space$ curves, which represent collision-free paths. In this way, sampling-based planners do not access the $C-space$ obstacles directly but only through the collision detector and the constructed data structure. When this level of abstraction is used, the planners are applicable to a wide range of problems by tailoring the collision detector to specific robots and applications.

4.6.1. The Probabilistic Road Map Method (PRM)

The probabilistic road map method (PRM) approach [51] attempts to approximate such a road map G in a computationally efficient way. The pre-processing phase of PRM, which can be extended to sampling-based road maps in general can be found in [51].

4.6.2. Rapidly Exploring Dense Trees (RDT)

The important property of these techniques is that the algorithm has to incrementally explore the properties of the $C-space$. An algorithm that accomplishes this objective is the rapidly exploring random tree (RRT) [52], which can be generalized to the rapidly exploring dense tree (RDT) for any dense sampling, deterministic or

random [53]. The main idea is to induce a Voronoi bias in the exploration process by selecting for expansion the point in the tree that is closest to $a(i)$ in each iteration. The probability that a vertex is chosen is proportional to the volume of its Voronoi region by using random samples. Several ways can be found in a planning algorithm in order to use RDTs. To develop a bidirectional search by growing two trees (one from each of q_I and $q(G)$ is a more efficient approach. To let the newest vertex of one tree be a substitute for $a(i)$ in extending the other is the simplest way to connect trees. Several works have extended, adapted, or applied RDTs in various applications like in [54].

4.7. Combined Formation Control and Path Planning

The objective of this section is to implement the potential field path planning methods to relative position based distributed formation control algorithms which are given in the previous sections. In order to find out the resulting effect of this combination, a convergence analysis will be given in detail.

4.7.1. Formation Control and Potential Field Path Planning

In this part, simulations are done using Webots Simulation Environment. Based on the algorithms given in Section 4.3, the potential field control is implemented for the four wheeled omnidirectional symmetric agents through the provided world.

The idea behind potential field is to create a field (or gradient) across the map that will direct the leader robot to the goal while avoiding the obstacle. This is possible by giving the goal an attractive force and obstacles very high repulsive force. Note that only the leader knows the whole environment, each follower just knows the positions of their neighbours. The position updates are provided by the supervisor after every time step, which is chosen as 10ms in this study. The simulation environment which includes obstacle, start and goal positions of agents respectively in turquoise, robot 1-leader in blue, followers are robot 2, 3 and 4 in red, black and magenta. The paths generated are shown with the same corresponding colors in Figure 4.9. Moreover the resulting equipotential contour plots of the obstacle and the goal for the leader are

given on the left upside.

The required parameters k_{att}, k_{rep} and ρ are chosen arbitrarly after trial and error stages as 0.6, 3 and 1. The initial relative distances between all agents are 2 meters. The initial positions and the formation are shown in Figure 4.8.

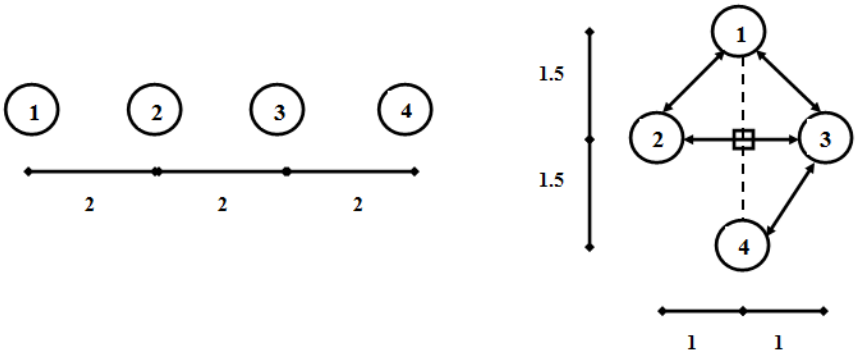

Figure 4.8. Initial positions and rhombus formation of all agents.

At the end of the simulation, the leader reaches the goal while avoiding the obstacle all the way through, whereas the followers reach the rhombus formation and maintain it until the end.

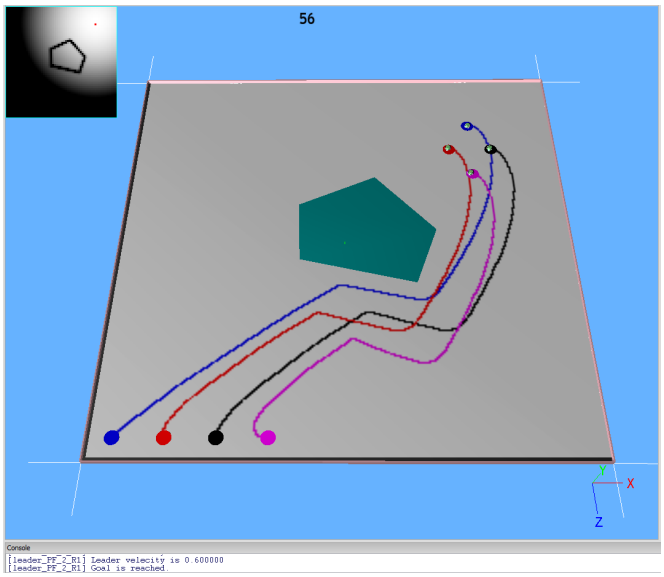

Figure 4.9. Leader based distributed formation control with obstacle avoidance and path planning.

In Figure 4.10, it can be seen that followers' velocities converge to the leader's velocity.

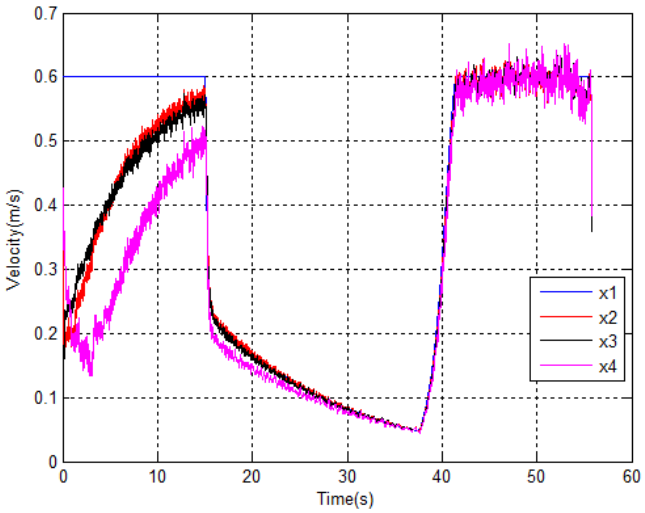

Figure 4.10. Velocity graph of agents.

In Figure 4.11, it can be seen that the relative distances between the leader and the followers converge to the desired rhombus formation distances.

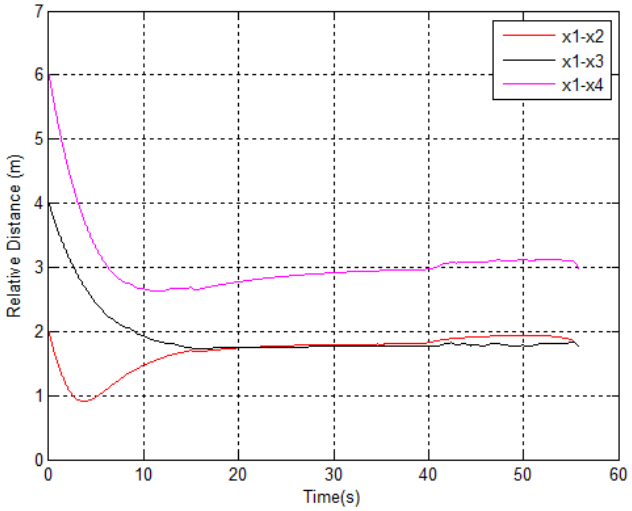

Figure 4.11. Relative distance graph between leader and followers.

4.8. Summary of the Chapter

In this chapter, before the combination of path planning with distributed formation control is examined, a detailed overview about concepts in motion planning algorithms is given. The potential field path planning algorithm is implemented for the leader agent, which is the only one knowing the whole environment whereas each follower just knows the position of its neighbours. Consequently, results of simulations done in Webots Environment show that agents reach indeed the desired rhombus formation while avoiding the obstacle. Finally, the followers' convergence analysis in terms of velocity and relative distances are given.

5. PERFORMANCE ANALYSIS OF CONSENSUS AND DISTRIBUTED FORMATION CONTROL ALGORITHMS

The previous chapters of this thesis consider consensus and formation that can be achieved in distributed multiple mobile robot systems where robots are assumed to be omnidirectional and with unconstrained velocity. However, mobile robots have usually a certain velocity and acceleration capacity that constrains them. That is why it is important to analyse the performance of the consensus and distributed formation control algorithms for different control inputs depending on the *maximum velocity* v_{max}. Since the distributed formation control algorithm is a generalization of the consensus algorithm, the issues will be evaluated in this chapter, first for consensus algorithm then for distributed formation control algorithm in terms of execution time and final situations.

5.1. Performance Analysis of Consensus Algorithm

We use in this section the control strategy for single integrator agents introduced in Section 2.3 and provide some complementary results. The distributed consensus algorithm in discrete-time model is

$$x_i(k+1) = x_i(k) + u_i(k) \tag{5.1}$$

where the control input $u_i(k)$ with step-size $\epsilon > 0$ is

$$u_i(k) = \epsilon \sum_{j=1}^{n} a_{ij}(x_j(k) - x_i(k)) \tag{5.2}$$

It is necessary to state the network configurations used in all different cases. The initial positions of the agents are $x_1 = (-2, 2)$, $x_2 = (-1.5, -1.5)$, $x_3 = (1.5, 0.5)$, $x_4 = (2, -2)$. Moreover, the possible flows of communication between the nodes are represented by arrows, expressing the edges of the graph in Figure 5.1.

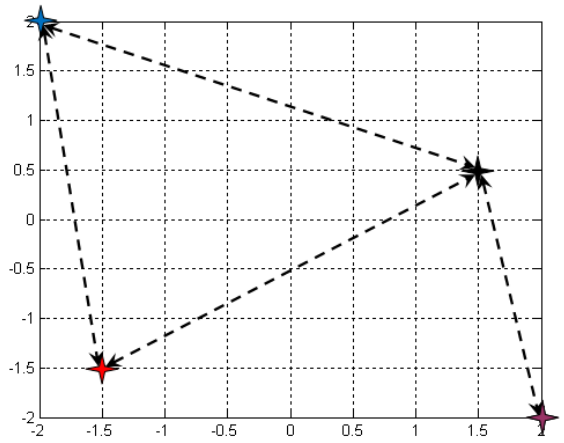

Figure 5.1. Graph representation of the network placed in the initial positions.

5.1.1. Proportional Control

In this case, the $u_i(k)$ value is taken as in Equation 5.3 for each agent i and step k.

$$u_i(k) = \epsilon \sum_{j=1}^{n} a_{ij}(x_j(k) - x_i(k)) \tag{5.3}$$

Figure 5.2 shows that the distributed consensus algorithm drives the agents to the common final location, at $(0, -0.25)$, the average consensus of the agents' initial positions. Let $\Delta d_i(k)$ be the distance to the next position for agent i at step k, the stopping criterion depending on the value of K representing the threshold for the simulations is chosen as

$$\Delta d_i(k) < K, \forall i \in \{1, ..., n\} \tag{5.4}$$

The execution time T_{exe} is equal to the total number of steps since the step duration is time unit, i.e.

$$T_{exe} = \sum_{k=1}^{k_1} 1 = k_1 \tag{5.5}$$

where k_1 is the total number of steps until consensus is reached. Figure 5.2 shows the corresponding output with $\epsilon = 0.1$, $K = 0.001$ which denotes the stopping criterion.

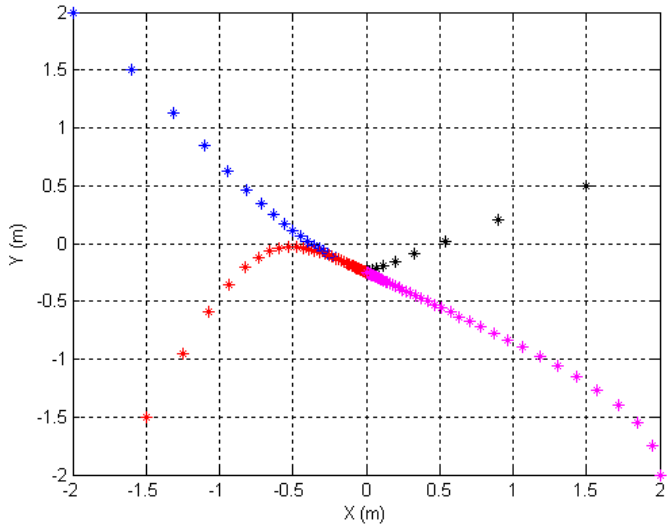

Figure 5.2. Simulation for Proportional Control with $T_{exe} = 55\,s$.

5.1.2. Saturated Control

In this case, we assume that the $u_i(k)$ of each agent depends on whether it is larger than v_{max} or not. In other words it is saturated, i.e.

$$u_i(k) = min\,\{\epsilon \sum_{j=1}^{n} a_{ij}(x_j(k) - x_i(k)), v_{max}\} \tag{5.6}$$

The execution time T_{exe} is equal to the total number of steps since the step duration is time unit. However, we observe that consensus is reached at another point depending on the value of v_{max}.

$$T_{exe} = \sum_{k=1}^{k_2} 1 = k_2 \tag{5.7}$$

where k_2 is the total number of steps until consensus is reached. Figure 5.3 shows different results with $\epsilon = 0.1$, $K = 0.001$. The consensus is located at: (a) $(-0.1223, 0.0191)$,

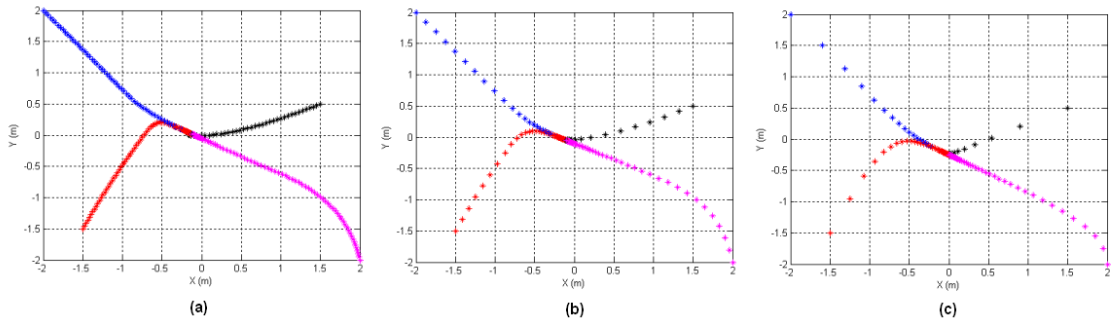

Figure 5.3. Simulations for Saturated Control with (a) $v_{max} = 0.05\,m/s$, $T_{exe} = 90\,s$ (b) $v_{max} = 0.2\,m/s$, $T_{exe} = 58\,s$ and (c) $v_{max} = 0.7\,m/s$, $T_{exe} = 55\,s$.

(b) $(-0.1050, -0.0479)$ and (c) $(0, -0.25)$.

5.1.3. Time-Improved Control

In this case, we assume that every agent in the network knows the distance $\Delta d_{max}(k)$ which denotes the largest distance to next position among all agents at each step k. Hence, it updates its $u_i(k)$ as

$$u_i(k) = \frac{\Delta d_i(k)}{\Delta d_{max}(k)} v_{max} \tag{5.8}$$

The execution time can be computed as

$$T_{exe} = \sum_{k=1}^{k_3} \frac{\Delta d_{max}(k)}{v_{max}} \tag{5.9}$$

where k_3 is the total number of steps until consensus is reached. As a result, the simulation remains the same as in the first case with Proportional Control. However, the execution time T_{exe} will change depending on the value of v_{max}. Figure 5.4 shows the corresponding output with $\epsilon = 0.1$, $K = 0.001$, and $v_{max} = 0.2\,m/s$.

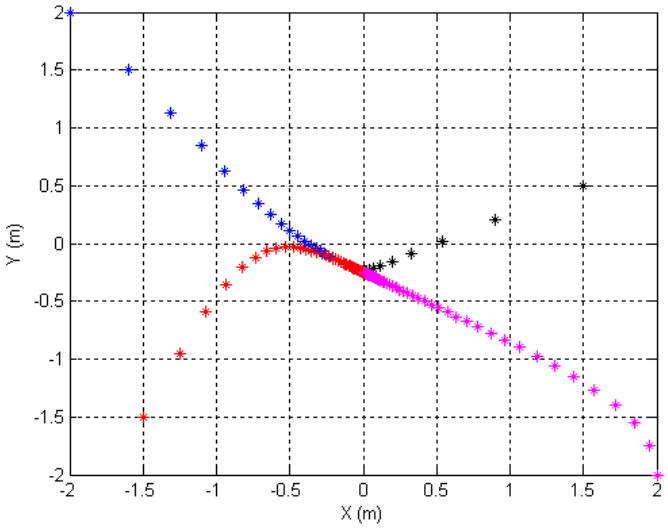

Figure 5.4. Simulation for Time-Improved Control with $T_{exe} = 18.2793\,s$ with $k_3 = 55$.

5.1.4. Maximum Velocity Control

In this case, we assume that every agent in the network updates its velocity to v_{max} in the same direction presented in the first case (Proportional Control) at each step. Figure 5.5 shows the corresponding output with $\epsilon = 0.1$ and $v_{max} = 0.2\,m/s$. As shown in Figure 5.5, there is an oscillation with a magnitude equal to v_{max} value. We have chosen the stopping criterion as the first oscillation observed.

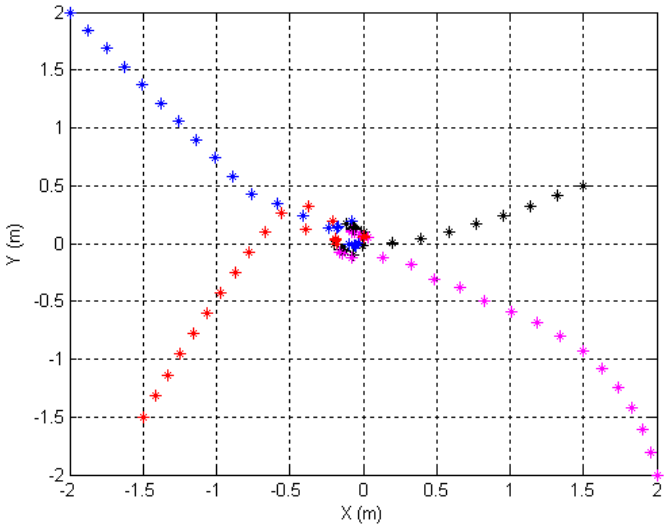

Figure 5.5. Simulation for Maximum Velocity Control with $T_{exe} = 28\,s$.

5.2. Performance Analysis of Distributed Formation Control Algorithm

In the previous section, performance analysis of consensus algorithm for four different control inputs have been carried out. We now investigate in this section the performances for distributed formation control algorithm as stated in Equation 5.10 with the same control inputs.

$$x_i(k+1) = x_i(k) + u_i(k) \tag{5.10}$$

where the control input $u_i(k)$, with step-size ϵ and desired inter-agent relative position between agent i and its neighbor j, r_{ij}, is

$$u_i(k) = \epsilon \sum_{j=1}^{n} a_{ij}(x_j(k) - x_i(k) - r_{ij}) \tag{5.11}$$

As presented before, Figure 5.6 shows the geometry of the desired formation to the network. The numerical examples illustrating these cases are given in the following subsections.

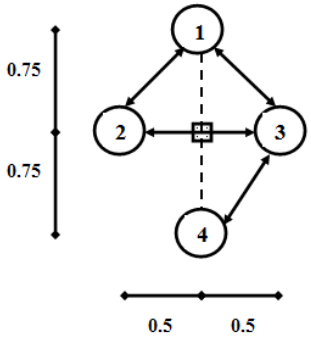

Figure 5.6. Target relative positions of the rhombus formation.

5.2.1. Proportional Control

In this case, the $u_i(k)$ value is taken as in Equation 5.12 for each agent i and step k.

$$u_i(k) = \epsilon \sum_{j=1}^{n} a_{ij}(x_j(k) - x_i(k) - r_{ij}) \tag{5.12}$$

Figure 5.7 shows that the algorithm drives the agents to the desired formation with $\epsilon = 0.1$, $K = 0.001$.

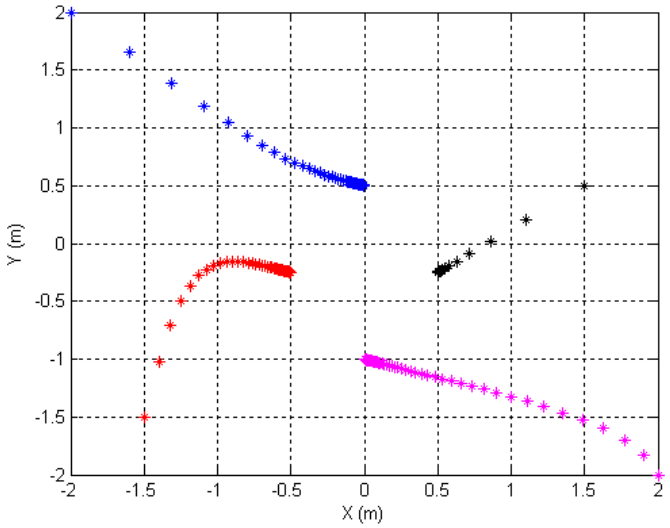

Figure 5.7. Simulation for Proportional Control with $T_{exe} = 54\,s$.

5.2.2. Saturated Control

In this case, we assume that the control input of each agent is:

$$u_i(k) = min\,\{\epsilon \sum_{j=1}^{n} a_{ij}(x_j(k) - x_i(k) - r_{ij}), v_{max}\}$$

We observed that formation is reached at other points depending on the value of v_{max}. Figure 5.3 shows different results with $\epsilon = 0.1$, $K = 0.001$.

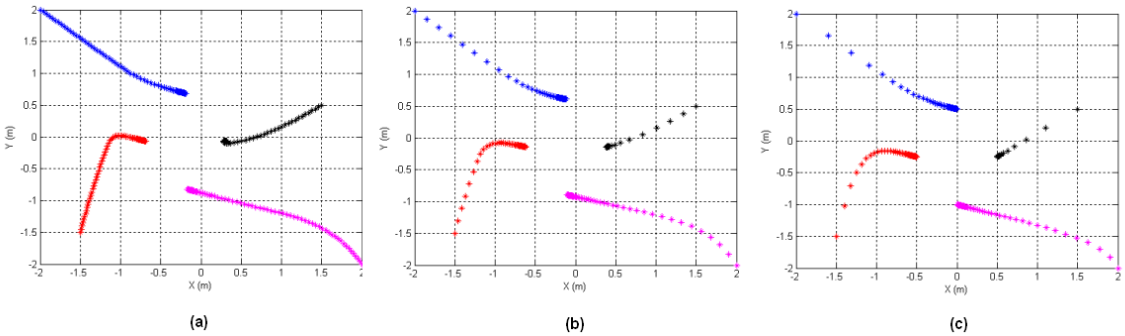

Figure 5.8. Simulations for Saturated Control with (a) $v_{max} = 0.05\,m/s$, $T_{exe} = 80\,s$ (b) $v_{max} = 0.2\,m/s$, $T_{exe} = 55\,s$ and (c) $v_{max} = 0.7\,m/s$, $T_{exe} = 54\,s$.

5.2.3. Time-Improved Control

In this case, each agent updates its $u_i(k)$ as

$$u_i(k) = \frac{\Delta d_i(k)}{\Delta d_{max}(k)} v_{max} \tag{5.13}$$

The simulation remains the same as in the first case (Proportional Control) however, the execution time T_{exe} will be changed depending on v_{max} value. Figure 5.9 shows the corresponding output with $\epsilon = 0.1$, $K = 0.001$, and $v_{max} = 0.2\,m/s$.

5.2.4. Maximum Velocity Control

In this case, we assume that every agent in the network updates its velocity to v_{max} in the same direction presented in the first case (Proportional Control) at each step. Figure 5.10 shows the corresponding output with $\epsilon = 0.1$ and $v_{max} = 0.2\,m/s$. As it is shown in Figure 5.10, there is an oscillation with a magnitude equal to v_{max} value. We have chosen the stopping criterion as the first oscillation observed.

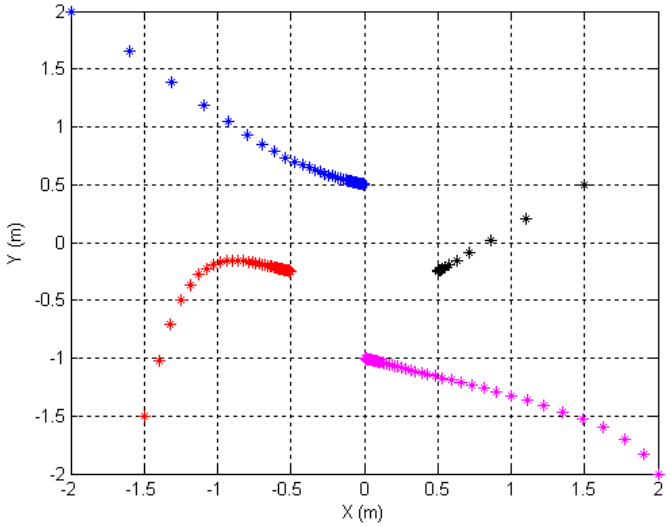

Figure 5.9. Simulation for Time-Improved Control with $T_{exe} = 15.1377\,s$ with $k_3 = 54$.

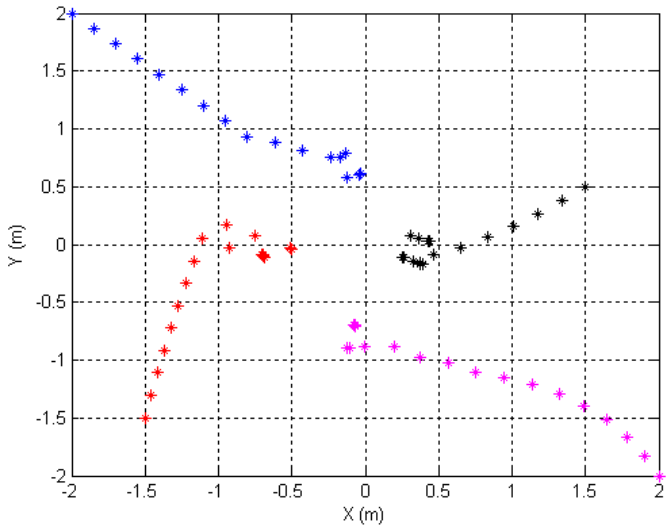

Figure 5.10. Simulation for Maximum Velocity Control with $T_{exe} = 25\,s$.

5.3. Summary of the Chapter

In this chapter, a performance analysis of consensus and distributed formation control algorithm has been carried out for four different control inputs: Proportional Control, Saturated Control, Time-Improved Control and Maximum Velocity Control. All parameters and results obtained in numerical examples for each case of the performance analysis of consensus are also valid for the performance analysis of distributed formation control.

As a result, we have seen that among these four different control inputs, Time-Improved Control Input is the one with the smallest execution time while the corresponding final state does not change. However, Maximum Velocity Control Input and Saturated Control Input have an effect on final states depending on the value of v_{max} which leads the final states to an oscillation or consensus at other points respectively. All simulations results are summarized in Table 5.1.

Table 5.1. Performance Evaluation of the Proposed Algorithms.

	CONSENSUS/FORMATION			
	Proportional Control	Saturated Control	Time-Improved Control	Maximum Velocity Control
Control Input ($u_i(k)$)	$\epsilon \sum_{j=1}^{n} a_{ij}(x_j(k) - x_i(k) - r_{ij})$	$min\{\epsilon \sum_{j=1}^{n} a_{ij}(x_j(k) - x_i(k) - r_{ij}), v_{max}\}$	$\frac{\Delta d_i(k)}{\Delta d_{max}(k)} v_{max}$	v_{max}
Final State	Consensus/Formation Reached	Consensus/Formation reached at another point/ other points	Consensus/Formation reached	Consensus/Formation oscillates
Execution Time (T_{exe})	$\sum_{k=1}^{k_1} 1 = k_1$	$\sum_{k=1}^{k_2} 1 = k_2$	$\sum_{k=1}^{k_3} 1 = \frac{\Delta d_{max}(k)}{v_{max}}$	$\sum_{k=1}^{k_4} 1 = k_4$
Stopping criterion	$\Delta d_i(k) < K, \forall i \in \{1, ..., n\}$	$\Delta d_i(k) < K, \forall i \in \{1, ..., n\}$	$\Delta d_i(k) < K, \forall i \in \{1, ..., n\}$	First observed oscillation

6. CONCLUSION

In this thesis, we adopted an algorithm used in consensus, and applied it in leader following relative position based distributed formation control for multi agent systems with obstacle avoidance and path planning. Furthermore we investigated its convergence properties both in theory and in simulation results.

The general literature review we presented showed the evolution in distributed formation control for multi agent systems. The need for the usage of motion planning in this area is clearly stated as well. Benefiting from graphical properties of networks and consensus algorithms, the combination of path planning with leader based distributed formation control, we have shown that the leader reaches the goal by generating a path and the followers reach a formation, maintain it while their velocities converge to the leader's.

As future work, different motion planning algorithms can be implemented in dynamically changing environments to be able to compare their advantages and disadvantages in execution time, energy consumption and efficiency.

REFERENCES

1. Reynolds, C. W., "Flocks, Herds, and Schools: A Distributed Behavioral Model", *Computer Graphics*, Vol. 21, No. 4, pp. 25-34, 1987.

2. Siciliano, B. and O. Khatip, *Handbook of Robotics*, Springer, New York, 2008.

3. Ren, W. and R. W. Beard, *Distributed consensus in multi-vehicle cooperative control : theory and applications*, Springer, London, 2008.

4. Nguyen, D. B., D. K. Do, "Formation Control of Mobile Robots", *International Journal of Computers, Communications & Control*, Vol. 1, No. 3, pp. 41-59, 2006.

5. Rodrigues, J., D. Figueira, C. Neves, and I. Ribeiro, "Leader-Following Graph-Based Distributed Formation Control" *Proceedings of Robotica, 8th Conference on Autonomous Robot Systems and Competitions*, pp. 71-78, 2008.

6. M. Egerstedt, "Graph-Theoretic Methods for Multi-Agent Coordination", *Proceedings of ROBOMAT*, pp. 3-12, 2007.

7. Olfati-Saber, R., J. A. Fax and R. M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems", *Proceedings of the IEEE*, Vol. 95, No. 1, pp. 215-233, 2007.

8. Olfati-Saber, R. and R. M. Murray, "Consensus Problems in Networks of Agents with Switching Topology and Time-delays", *IEEE Transactions on Automatic Control*, Vol. 49, No. 9, pp. 1520-1533, 2004.

9. Fax, J. A. and R. M. Murray, "Information Flow and Cooperative Control of Vehicle Formations", *IEEE Transactions on Automatic Control*, Vol. 49, No. 9, pp. 1465-1476, 2004.

10. DeGroot, M. H., "Reaching a Consensus", *Journal of American Statistical Association*, Vol. 69, No. 345, pp. 118-121, 1974.

11. Borkar, V. and P. Varaiya, "Asymptotic Agreement in Distributed Estimation", *IEEE Transactions on Automatic Control*, Vol. 27, No. 3, pp. 650-655, 1982.

12. Jadbabaie, A., J. Lin, and A. S. Morse, "Coordination of Groups of Mobile Autonomous Agents using Nearest Neighbor Rules", *IEEE Transactions on Automatic Control*, Vol. 48, No. 6, pp. 988-1001, 2003.

13. Moreau, L., "Stability of Multi-agent Systems with Time-dependent Communication Links", *IEEE Transactions on Automatic Control*, Vol. 50, No. 2, pp. 169-182, 2005.

14. Ren, W. and R. W. Beard, "Consensus Seeking in Multiagent Systems under Dynamically Changing Interaction Topologies", *IEEE Transactions on Automatic Control*, Vol. 50, No. 5, pp. 655-661, 2005.

15. Ren, W., R. W. Beard, and T. W. McLain, "Coordination Variables and Consensus Building in Multiple Vehicle Systems", in *Cooperative Control: A Post-Workshop Volume 2003 Block Island Workshop on Cooperative Control*, V. Kumar, N.E. Leonard, and A.S. Morse, Eds. New York: Springer-Verlag, pp. 171-188, 2005.

16. Dimarogonas, D. V. and K. J. Kyriakopoulos, "On the Rendezvous Problem for Multiple Nonholonomic Agents", *IEEE Transactions on Automatic Control*, Vol. 52, No. 5, pp. 916-922, 2007.

17. Gustavi, T., D. V. Dimarogonas, M. Egerstedt and X. Hu, "Sufficient Conditions for Connectivity Maintenance and Rendezvous in Leader-follower Networks", *Automatica*, Vol. 46, No. 1, pp. 133-139, 2010.

18. Horn, R. A. and C. R. Johnson, *Matrix Analysis*, Cambridge, U.K. : Cambridge Univ. Press, 1987.

19. Lin, J., A. S. Morse, and B. D. O. Anderson, "The Multi-agent Rendezvous Problem", *Proceedings of the IEEE Conference on Decision and Control*, pp. 1508-1513, Maui, HI, 2003.

20. Martinez, S., J. Cortes, and F. Bullo, "On Robust Rendezvous for Mobile Autonomous Agents", *IFAC World Congress*, Prague, Czech Republic, 2005.

21. Ren, W., "Consensus Strategies for Cooperative Control of Vehicle Formations", *IET Control Theory & Applications*, Vol.1, No.2, pp. 505-512, 2007.

22. Olfati-Saber, R., "Flocking for Multi-agent Dynamic Systems: Algorithms and theory", *IEEE Transactions on Automatic Control*, Vol. 51, No.3, pp. 401-420, 2006.

23. Tanner, H. G., A. Jadbabaie, and G. J. Pappas., "Flocking in Fixed and Switching Networks", *IEEE Transactions on Automatic Control*, Vol. 52, No. 5, pp. 863-868, 2007.

24. Casbeer, D. W., D. B. Kingston, R. W. Beard, T. W. McLain, S. M. Li, and R. Mehra, "Cooperative Forest Fire Surveillance using a Team of Small Unmanned Air Vehicles", *International Journal of Systems Sciences*, Vol. 37, No. 6, pp. 351-360, 2006.

25. Olfati-Saber, R. and J. S. Shamma, "Consensus Filters for Sensor Networks and Distributed Sensor Fusion", *Proceedings of the IEEE Conference on Decision and Control, and the European Control Conference*, pp. 6698-6703, Seville, Spain, 2005.

26. Spanos, D. P. and R. M. Murray, "Distributed Sensor Fusion using Dynamic Consensus", *IFAC World Congress*, Prague, Czech Republic, 2005.

27. Ji, M. and M. Egerstedt, "Distributed Coordination Control of Multi-agent Systems while Preserving Connectedness", *IEEE Transactions on Robotics*, Vol. 23, pp. 693-703, 2007.

28. Godsil, C. and G. Royle, *Algebraic Graph Theory*, Springer-Verlag, New York, 2001.

29. Ferrari-Trecate, G., M. Egerstedt, A. Buffa, and M. Ji, "Laplacian Sheep: A Hybrid, Stop-Go Policy for Leader-Based Containment Control", *Hybrid Systems: Computation and Control*, pp. 212-226, Springer-Verlag, Santa Barbara, CA, 2006.

30. Axelsson, H., A. Muhammad, and M. Egerstedt, "Autonomous Formation Switching for Multiple, Mobile Robots", *IFAC Conference on Analysis and Design of Hybrid Systems ADHS03*, pp. 147-152, Saint-Malo, Brittany, France, 2003.

31. Fax, J. A. and R. M. Murray, "Graph Laplacian and Stabilization of Vehicle Formations", *IFAC 15th Proceedings*, pp. 283-288, 2002.

32. Lin, Z., M. Broucke, and B. Francis, "Local Control Strategies for Groups of Mobile Autonomous Agents", *IEEE Transactions on Automous Control*, Vol. 49, No. 4, pp. 622-629, 2004.

33. Ando, H., Y. Oasa, I. Suzuki, and M. Yamashita, "Distributed Memoryless Point Convergence Algorithm for Mobile Robots with Limited Visibility", *IEEE Transactions on Robotics Automation*, Vol.15, No. 5, pp. 818-828, 1999.

34. Notarstefano, G., K. Savla, F. Bullo, and A. Jadbabaie, "Maintaining Limited-Range Connectivity among Second-order agents", *Proceedings of American Control Conference*, pp. 2124-2129, 2006.

35. Couzin, I., J. Krause, N. Franks, and S. Levin, "Effective Leadership and Decision Making in Animal Groups on the Move", *Nature, 433*, pp. 513-516, 2005.

36. Cihan, O., *Distributed Synchronization in Delayed and Topology Varying Networks*, M.S. Thesis, 2009.

37. Boyd, S., P. Diacounis, and L. Xiao, "Fastest Mixing Markov Chain on A Graph",

SIAM Review, Vol. 46, pp. 667-689, 2004.

38. Lin, Z., B. Francis, and M. Maggiore, "Necessary and Sufficient Graphical Conditions for Formation Control of Unicycles", *IEEE Transactions on Automation Control*, Vol. 50, No.1, pp. 121-127, 2005.

39. Caughman, J. S., G. Lafferriere, J. J .P. Veerman, and A. Williams, "Decentralized Control of Vehicle Formations", *System & Control Letters*, Vol. 54, No. 9, pp. 899-910, 2005.

40. Akar, M., Ö.F. Varol, F. İleri, H. Esen, R. S. Kuzu and A. Yurdakurban, *BRocks 2010 Team Description*, 2010, http://small-size.informatik.uni-bremen.de/robocup2010:teams/, accessed at May 2011.

41. Rojas, R. and A. G. Förster, "Holonomic control of a robot with an omnidirectional drive", *Künstliche Intelligenz, Böttcher IT*, Vol. 20, No. 2, pp. 12-17, Verlag, 2006.

42. Pugh, J. and A. Martinoli, "Small-Scale Robot Formation Movement Using a Simple On-Board Relative Positioning System", *International Symposium on Experimental Robotics*, 2006.

43. Szabo, R. "Topological Navigation of Simulated Robots using Occupancy Grid", *International Journal of Advanced Robotic Systems*, Vol. 1, No. 3, pp. 201-206, 2004.

44. Sariel, S., B. Tucker, and N. Erdogan, "Incremental Multi-Robot Task Selection for Resource Constrained and Interrelated Tasks", *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems San Diego*, CA, USA, 2007.

45. Cyberbotics Ltd., 2011, http://www.cyberbotics.com/, accessed at May 2011.

46. Latombe, J. C., *Robot Motion Planning*, Kluwer Academic Publishers, Norwood, MA, 1991.

47. Russell, S., and P. Norvig, *Artificial Intelligence, a Modern Approach*, Prentice

Hall International, New Jersey, 1995.

48. Latombe, J. C. and J. Barraquand, "Robot Motion Planning: A Distributed Presentation Approach", *International Journal of Robotics Research, 10*, pp. 628-649, 1991.

49. Siegwart, R. and I. R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*, The MIT Press, Cambridge, MA, 2004.

50. Jacobs, R. and J. Canny, "Planning Smooth Paths for Mobile Robots", *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 2-7, 1989.

51. Kavraki, L. E., P. Svestka, J. C. Latombe and M. H. Overmars, "Probabilistic Road Maps for Path Planning in High-dimensional Configuration Spaces", *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 4, pp. 566-580, 1996.

52. LaValle, S. M. and J. J. Kuffner, "Rapidly-exploring Random Trees: Progress and Prospects", *Algorithmic and Computational Robotics: New Direction, ed. By Donald, B.R., Lynch, K.M. and Rus D.(A. K. Peters,Wellesley 2001)* pp. 293-308, 2001.

53. LaValle, S. M., *Planning Algorithms*, Cambridge University Press, Cambridge 2006.

54. Bruce, J. and M. Veloso, "Real-time Randomized Path Planning for Robot Navigation", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2383-2388, 2002.