Multi-agent Communication Protocols with Emergent Behaviour

by

Ghada A.K. Al-Hudhud

A dissertation submitted in partial fulfilment of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

www.dmu.ac.uk
DE MONTFORT UNIVERSITY
LEICESTER • BEDFORD

DE MONTFORT UNIVERSITY
at
LEICESTER

2005

Abstract

The emergent behaviour of a multiagent system depends on the component agents and how they interact. A critical part of interaction between agents is communication. This thesis presents a multi-agent system communication model for physical moving agents. The work presented in this thesis provides all the tools to create a physical multi-agent communication system. The model integrates different agent technologies at both the micro and macro level. The micro structure involves the architecture of the individual components in the system whilst the macro structure involves the interaction relationships between these individual components in the system.

Regarding the micro structure of the system, the model provides the description of a novel hybrid BDI-Blackboard architectured agent that builds-in a hybrid of reactive and deliberative agent. The macro structure of the system, provided by this model, provides the operational specifications of the communication protocols. The thesis presents a theory of communication that integrates an animal intelligence technique together with a cognitive intelligence one. This results in a local co-ordination of movements, and global task co-ordination. Accordingly, agents are designed to communicate with other agents in order to coordinate their movements via a set of behavioural rules. These behavioural rules allow a simple directed flocking behaviour to emerge. A flocking algorithm is used because it satisfies a major objective, i.e. it has a real time response to local environmental changes and minimises the cost of path planning. A higher level communication mechanism is implemented for task distribution that is carried out via a blackboard conversation and negotiation process with a ground based controller. All the tasks are distributed as team tasks. A novel utilization of speech acts as communication utterances through a blackboard negotiation process is proposed.

In order to implement the proposed communication model, a virtual environment is built that satisfies the realism of representing the agents, environment, and the sensors as well as representing the actions. The virtual environment used in the work is built as a semi-immersive full-scale environment and provides the visualisation tools required to test, modify, compare and evaluate different behaviours under different conditions. The visualization tools allow the user to visualize agents negotiations and interacting with them. The 3D visualisation and simulation tools allow the communication protocol to be tested and the emergent behaviour to be seen in an easy and understandable manner. The developed virtual environment can be used as a toolkit to test different communication protocols and different agent's architecture in real time.

To....

My mother....

and my late father....

Acknowledgements

I would like to express my special thanks to my major professor, Dr. Martin Turner. His sound advice has been my beacon throughout my stay at De Montfort University. From him, I learnt how to take an idea from its abstract infancy, give it shape and put it down effectively in operation. I hope to achieve his level of organization and way of thinking one day. I would also like to thank Dr. Aladdin Ayesh for being patient with my numerous questions and queries. He made time from his busy schedule to help me get the results I needed to complete this work. Many thanks to Mr. Howell Istance whose critical eye, and enlightened mentoring were instrumental and inspiring. I will always remember the lively discussions we had. I would like also to thank Mr. Bryan Bramer for his assistance in programming the network phase of the work. A special thank for Professor Gwin Evans for reading the final version of this thesis and for the invaluable suggestions. Not forgotten to thank my colleague Simon Coupland for the spirit of friendship and mutual assistance.

I express my sincere gratitude to my parents, for supporting my learning interests through all my life. They made sure that my brothers, sisters and I had the best of opportunities growing up.

Last in this list but first in my heart, sincere thanks are due to my husband *Mohammed* for his love and fabulous encouragement, which I needed it indeed, during my study. Also, my great love to my children: *Malik*, *Ahmed*, *Bilal*, and *Minas*, for their patience and emotional support.

Symbols and Abbreviations

Abbreviations

MAS	Multi-Agent System
$PMAS$	Physical Multi-Agent System
$MMRS$	Multiple Mobile Robot System
VE	Virtual Environment
$LSMAS$	Large Scale Multi-Agent System
$BB-N$	Blackboard Negotiation
SA	Speech Act
$AKRS$	An Agent's Knowledge Representational System
CU	Cognitive unit
SOM	State of Mind
$PTA-agent$	Perceive-Think-Act agent
$HBDI$ Architecture	Hybrid Belief-Desire-Intention architecture
$HBDIB$ Architecture	Hybrid BDI-Blackboard architecture
$LC-Model$	Local Communication Model
$GC-Model$	Global Communication Model
$LGC-Model$	Local Global Communication Model
GBC	Ground based controller
FOV	Filed of view
ρ	Ratio= $\frac{Team\ Size}{NumberofTeams}$

Flocking Rule Variables

A_i	Agent i
P_{A_i}	Current position of agent A_i
Ψ_{A_i}	The current heading of the agent A_i
SR	Sensor Range
S_d	Minimum separation distances allowed between objects
R_α	Alignment Rule
$C_{A_i}^{R_\alpha}$	The centroid of alignment rule α for the agent A_i
$w_{A_i}^{R_\alpha}$	The weight of alignment rule R_α for the agent A_i
α_{A_j}	The nearest neighbour's $(A_j's)$ heading angle
$\theta_{A_j}^{R_\alpha}$	The correction angle produced by the alignment rule for agent A_j
R_β	Cohesion Rule
$C_{A_i}^{R_\beta}$	The centroid of cohesion rule R_β for the agent A_i
$w_{A_i}^{R_\beta}$	The weight of cohesion rule R_β for the agent A_i
β_{A_j}	The direction of the cohesion centroid for agent A_i
$\theta_{A_j}^{R_\beta}$	The correction angle produced by cohesion rule R_β for agent A_i
R_γ	Collision Avoidance Rule
$C_{A_i}^{R_\gamma}$	The centroid of collision avoidance rule for the agent A_i
$w_{A_i}^{R_\gamma}$	The weight of the collision avoidance rule for the agent A_i
$\theta_{A_j}^{R_\gamma}$	The correction angle produced by the collision avoidance rule R_γ for agent A_i

Blackboard Variables

R_{BB}	Blackboard Rule
Pos_{tar}	Position of the target issued by the GBC
$Dist_{A_i}^{tar}$	Distance form an agent A_i to the target tar
$Pos_{A_i}^{start}$	The Agent's position when it started the task
$\theta_{A_j}^{R_{BB}}$	The $(A_j's)$ The correction angle produced by BB rule
$C_{A_i}^{R_{BB}}$	The centroid of BB rule for the agent A_i
$w_{A_i}^{R_{BB}}$	The blackboard rule weight.

Contents

Symbols and Abbreviations **v**

List of Tables **xiii**

List of Figures **xv**

1 Introduction **1**

1.1 Implementation of a MAS Model . . . 3

1.1.1 Methodologies . . . 3

1.1.2 Task Level Techniques in a $PMAS$ . . . 4

1.1.3 Visual Simulation of a physical MAS . . . 5

1.2 Main Contribution of the Thesis . . . 6

1.3 Structure of the Thesis . . . 7

2 Theoretical Background **9**

2.1 Common Communication Techniques within Multi-Agent Systems MASs . 10

2.1.1 Communication via Negotiation . . . 11

2.1.2 Communication via Speech Acts (SA) . . . 13

2.1.3 Communication via a Joint Action . . . 16

2.1.4 Communication via Social Behaviours . . . 17

2.2 Common Agent's Architectures . . . 19

2.2.1 Hybrid Belief-Desire-Intention Architecture . . . 20

2.2.2 Blackboard Architecture . . . 21

2.3 Common Simulation and Visualisation Tools 22
2.3.1 Why 3D simulations 22
2.3.2 A Comparison between 2D and 3D-Simulations 23
2.3.3 Current implementation of VEs 25
2.4 Hardware Implementation 27
2.4.1 Navigation 28
2.4.2 Perception 30
2.5 Summary 32

3 Computational Framework 36
3.1 Micro Structure of the MAS 38
3.1.1 The Agent's Architecture 38
3.2 An Agents Knowledge Representational System 40
3.2.1 An Agents Sources of Knowledge and Information Management . . . 42
3.2.2 The Agents Sensor and Sensing Strategies 44
3.3 Macro Structure of the MAS 45
3.3.1 Local Interaction Technique: Flocking Algorithm 46
3.3.2 Global Communication Technique: Blackboard Negotiation 51
3.4 Communication Algorithm 53
3.5 Communication Protocols 55
3.5.1 Encoding an Agent's Knowledge into Communicative Acts 55
3.5.2 An Example of an Agent's Conversation: Speech Acts in a Sheet Form 57
3.5.3 Modes of Communication Specified by the Proposed Protocol 60
3.6 Conclusion 62

4 Visualizing Agents' Negotiations 66
4.1 A 3D Representation of the MAS model 68
4.1.1 3D Representation of the Agents and Environment: Geometrical Realism 68
4.1.2 Simulating an Agent's Artificial Vision System 70
4.1.3 Simulating Agents' Interactions: Behavioural Realism 73
4.2 Simulating a MAS Communication model within a Semi-Immersive VE . . 75

4.2.1 The Desktop VE 75
4.2.2 The Large Scale VE 76
4.2.3 User's Interaction with the Agents in a VE 76
4.2.4 Advantages of a Large Scale Simulation 78
4.3 Evaluating the Emergent Behaviours: Visual Analysis 81
4.3.1 Local Communication Model ($LC - Model$) 84
4.3.2 Global Communication Model ($GC - Model$) 86
4.3.3 Local-Global Communication Model ($LGC - Model$) 89
4.3.4 Oscillate State Detection and Follow Wall Mode: $FW - Model$ . . . 92
4.4 Conclusion 93

5 Quantitative Analysis 95
5.1 Scalability 96
5.2 Analysing Emergent Behaviour for the Flocking System 98
5.2.1 Advantages of Flocking System: Routes and Coverage Areas 98
5.2.2 Sensor Range SR 102
5.2.3 Controlling the Interaction Weights of the Agent's Subsystems . . . 106
5.2.4 Detecting Locations of Complexity within the Environment 113
5.3 Grouping Technique: Optimum Team Size 118
5.4 Discussion 124

6 The Design of a Co-operative Multi-Robots Team 127
6.1 Operational Specifications 128
6.1.1 Hardware/Software Specifications 128
6.1.2 Network Protocols Specification 130
6.2 Experimentation 132
6.3 Discussion 136

7 Conclusion 139
7.1 Interactive Communication within a $PMAS$ Model 139
7.2 A $HBDIB$ Architecture 140
7.3 The VE as a Testbed 141

7.4 Verification 142
7.5 Implementation in Real Mobile Robots 143
7.6 Summary and Future Work 143

Bibliography **145**

A Conventional Implementation of Flocking Algorithms **160**
A.1 Experiment 160
A.2 Recommendation and Summary 161

B Developing a 3D Simulator **165**
B.1 Software Development 165
B.2 The Runtime Loop 166
B.2.1 User Defined Application class derived from VegaPrime Application 167
B.2.2 User Defined Method 167
B.2.3 Configure Method: configure() 168

C Quantitative Assessments **170**
C.1 Sensor Range 170
C.2 Controlling the Interaction Weights 172
C.3 Optimum Team Size 173

D Network Multicasting Protocol **175**
D.1 Why Use a Linux Based Operating System 175
D.2 IP Network Settings 176
D.3 Programming 178
D.3.1 A Multirobot Chat Server 178
D.3.2 Reading the User Inputs: User is a Client 179
D.3.3 Robot's are Other Clients 181

E Publications **185**

F Software System & Demos **187**

List of Tables

3.1 An agent's information management and sources of knowledge. 43
3.2 Perception zone for the three flocking rules 47
3.3 SABN- operators and predicates 57
3.4 (GBC) is delegating a task to a set of agents A_i where $i = 0, ..., n$ 58
3.5 Modes of Communication specified by the proposed protocol 61
3.6 The communication algorithm. 65

5.1 Scalability. 97
5.2 Completion time in terms of the sensor range using different S_d values for 20 agents. 105
5.3 The values of set of interaction weights 107
5.4 The interaction weights, modified. 110
5.5 The completion time for the first, 50%, 100% of arrivals for a set of 20 agents.113
5.6 The possible combinations, and resulting ratios ρ for a set of 48 agents. . . 119
5.7 The effect of varying ρ on the arrival rate, see table 5.6 for the ratio. 121

B.1 Vega Prime Run Time Loop 169

C.1 Completion time in terms of the sensor range and Sd values for 5 agents. . 171
C.2 Completion time in terms of the sensor range and Sd values for 45 agents. . 172
C.3 The completion time for the first, 50%,100% of arrivals for a set of 20 agents, $Coh_w <= 1$, see table 5.10. 173

C.4 The completion time for the first, 50%,100% of arrivals for a set of 20 agents, $Coh_w < 1$, see table 5.11. 173

C.5 The effect of varying ρ on the completion time for first arrival, 50%, 100% arrivals for a set of 36 agents. 174

D.1 User's Main 182

D.2 The Action Class 183

D.3 The Robot's Main Function 184

List of Figures

2.1 High level abstract system structure 35

3.1 The agent's architecture 39
3.2 The agent's mind 41
3.3 The flocking rules 50

4.1 A 3D representation of an agent. 69
4.2 A 3D representation of the environment, a simple maze. 70
4.3 The simple maze with some extra obstacles. 71
4.4 Rotating Sensor. 72
4.5 The sensor renders green once it hits an object. Object A is detected while object B is not. 73
4.6 A screenshot, photograph taken while running a large scale system, demonstration to a group of students. 77
4.7 Virtual laboratory with different starting points and target positions marked. 78
4.8 Viewing agents' interaction; zoom in. 79
4.9 Team members split into teams; zoom out. 80
4.10 The large scale simulation supports real physical sizes and spaces. 81
4.11 The different scales give different representation of the same spaces. 82
4.12 Leicester Reality Centre, De Montfort University, displaying multiple views simultaneously. 83
4.13 Viewing a large number of agents, photograph taken while running the large scale system. 84

4.14 A set of agents are moving within a team and influenced by the cohesion force and the alignment force. 85
4.15 A team of agents moving forward, the team members split into two teams as they encountered the wall. 86
4.16 Viewing agents' interaction in a large scaled model. 87
4.17 Viewing agents' interaction in small scaled model. 88
4.18 Team members are circling around a target position. 90

5.1 The computation time as a function of the population size. 98
5.2 The computation time as a function of the population size up to 100 agents, note the non-linearity as the population size becomes bigger than 50 agents. 99
5.3 The agents select different route whilst they are moving in a team. 100
5.4 The flocking behaviour supports maximising the coverage area. 101
5.5 The covered area is less with the flocking system switched off. 102
5.6 Number of frames as a function of sensor range. 106
5.7 The smaller sensor range reduces the influence of the flocking system. . . . 107
5.8 A larger sensor range increases the influence of the flocking system. 108
5.9 The number of movements towards the target increase as a result of a higher influence of the flocking system. 109
5.10 The interaction weights over the first 200 frames. The cohesion weight dominates the interaction weights whenever the avoidance weight is zero. The unmodified cohesion weight values are shown in table 5.3. 111
5.11 The interaction weights over the first 200 frames, with the cohesion weight modified according to the values shown in (table 5.4). 112
5.12 The completion time versus number of arrivals for a set of 20 agents, which shows the effect of reducing the cohesion weight, table 5.5 113
5.13 The distribution of the cohesion weights associated with the cohesion forces for each frame. 114
5.14 The individual deviations from the mean positions for the set of 5 agents. . 116
5.15 The average standard deviation for the same group, the set of 5 agents. . . 118
5.16 Variation in first arrival time, fifty per cent, and completion time vs ratio. . 120
5.17 The uniformity of arrivals with respect to different ratios. 122
5.18 Optimum Team size, as a function of ratio for 36 agents. 123
5.19 Optimum Team size, a uniform arrival rate with 9 teams of 36 agents. . . . 124

6.1 The robots with, although, different platforms, sonar sensors and little antenna for the wireless connection. 128

6.2 The robots recieve the user's commands as well as messages from other robots via the ethernet connection. 130

6.3 Experiment set up. Top, a maze; the environment where the robots operate. Bottom, the maze . with two removable obstacles. 133

6.4 The average standard deviation, distance form the mean position for the two robots 135

6.5 The two robots in operation. 138

A.1 Left, a randomly initiated set of agents. Right,cohesion force binds each team members together. 162

A.2 Left, cohesion force binds the team members together after steering around the obstacle. Right, a screenshot shows both the alignment force and the collision avoidance rule. 163

A.3 Screenshots shows how the team members split into two teams to avoid colliding with the obstacle.Left, black-team members are split into two teams whilst avoiding the obstacle. Right, white team members are split into teams, note the two agents from the red team are trying to avoid hitting each other. 164

A.4 Flocking Rules. 164

B.1 Vega Prime System Architecture, [5]. 166

C.1 The completition time as a function of sensor range for set of 5 agents. . . . 171

C.2 The completition time as a function of sensor range for set of 45 agents. . . 172

Chapter 1

Introduction

Robotic applications have been extensively studied, in particular, for carrying out tasks that reduce human presence in certain or dangerous tasks, increase productivity, and reduce the time cost. Examples of these tasks include using multiple mobile robots system ($MMRS$) for surface planetary exploration, cleaning toxic wastes, fire extinguishing, or deep sea diving tasks. In these tasks, work can be performed more efficiently and reliably using several co-operative robots [47].

Specifying the type of robots to be used is considered essential to obtain co-operation. Freitas *et al* [39] believe that co-operative robots must be autonomous. Autonomy is important because operating these robots remotely from a static station can be time consuming especially in the cases where the robots encounter dangerous situations. In addition, Freitas *et al* state that these robots must possess a sufficient level of intelligence. Levels of intelligence could vary depending on whether it is the intelligence of the individuals that is required in the system or it is the system intelligence.

Currently, these levels of intelligence are widely studied in the field of communication within Multi-Agent Systems (MASs). The common features of MASs are intelligence, autonomy, parallel processing, interaction through communication and co-operation. Recently, MASs have been considered an efficient tool for modelling the system intelligence and the corresponding emergent behaviour. In addition, the recent advances in technologies, methods, and theories of agents and (MASs) are currently contributing to diverse domains. Among these domains are the multiple mobile robot systems ($MMRS$s) domain which exemplifies the main features of the multi-agent systems and are considered as an implementation of a Physical Multi-Agent System ($PMAS$).

Unlike the conventional AI modelling technique that requires the individual entities in the system to be highly intelligent, modelling a MAS requires that each individual agent must possess a high level of an interaction mechanism which when implemented generates global behaviour. Methods to generate global behaviours from many local behaviours within MASs have been applied to a varied sets of fields. An example is presented by Reynolds in [81] as a mathematical simulation of flocking behaviour. This example implements an adaptable motion technique to co-ordinate the movements of a group of agents. Another example is presented by London [66] that introduced a new MAS model of price dynamics by considering some simplified cases to model the complexity and criticality in financial time series. Schlecht and Joseph also presented an example in [83], that implemented emergent behaviour techniques for modelling mission planning for unmanned air vehicles.

1.1 Implementation of a MAS Model

The implementation of any MAS model implies putting the theoretical aspects (i.e computing architecture of each entity, methodology and techniques), and the suitable visual simulation tools together in operation. This implies specifying the interaction mechanisms to produce a behavioural model, depending on the implementation area, modelling the corresponding agent's architecture that meets the behavioural model, and modelling the agent's actions. Finally, it is essential to specify the simulation tools that support the specific of the application. In addition, these simulation tools must enable the user to visualise the agents emergent behaviour in order to assess the efficiency of a MAS model.

1.1.1 Methodologies

Despite the possible benefits of multiplicity, some problems are now introduced, for example path planning for a group of mobile robots. A cleaning task, presented in [68], requires robots to have a pre-defined path plan to cover the unoccupied areas in a specified environment. For relatively small scaled systems, this has been shown to be feasible, whilst it is impractical to pre-specify the paths for a large number of agents. Some examples of the systems which are capable of complex path planning are presented in [100], [60], and these systems require an accurate environment-map before the path planning algorithm is executed.

For a dynamically changeable environment with moving agents, it is simply not practical to try to fully specify paths for all objects nor to provide an accurate map in advance.

Centralised solutions for path planning, of which an example is presented in [42], are also impractical and are usually highly unscalable. An adoptive movement technique is required to co-ordinate movements of a large group in a dynamic environment. An example of a system that uses an adaptable motion co-ordination technique is presented by Tang in [92], where an A^* path planning algorithm is used for searching an obstacle free path with the least cost for the agents from a starting point to the goal point. In addition, the dynamic changes in the environment leads to the necessity to recalculate the navigation path. In the above examples the agents' new decisions are dependent on the previous actions, i.e. the agents still need to calculate the most economic path.

1.1.2 Task Level Techniques in a $PMAS$

The examples presented in the previous section have shown that simulating movement plans is sometimes feasible without assigning any higher specific tasks. For the proposed research, we need also a task assignment via a communication solution to be added. Previous methods, an example is presented in [6], addressed a lack of interactive task allocation algorithms as a consequent of the lack of sufficient communication. Another example is presented in [53] which is considered computationally intensive and therefore unsuitable for real time purposes.

The communication solution must measure the amount of communication required in order to control the agents' actions, whether these are movements or transmitting or receiving information. In addition, the way these agents communicate must be adaptable to the changes in the environment in which they are situated. Therefore, in order to overcome the above highlighted problems, a communication protocol is proposed that discards pre-specification and uses a mobile coordination and communication techniques and results in suitable response actions which depends on decision-making algorithms while the agent is in motion.

1.1.3 Visual Simulation of a physical *MAS*

Visualising the agents' behaviours forms an essential stage of implementing any *MAS*. This implies that the simulation tools must give sufficient and realistic representation of the agents and their behaviours in a simulated world. Many current simulations use either 2D representation tools, as JAVA platforms, or 3D representations that provid only the minimal requirements of representing shapes and actions. Realistic representations also implies that the simulation tools must support the specifics of the application, e.g. these tools must enable the user to simulate the real-like sensors for the physical robots. This helps the user to easily understand the theoretical aspects of the model and provides the user with a clear idea about the possible problems of the model as well as possible solutions.

The realism of the simulation requires real time simulation tools, i.e. no pre-computations. Loscos *et al* [67] described a crowd behaviour real time simulation that allowed for the rendering of the scene and the individuals as well as simulating the crowd behaviours. The system described by Loscos *et al* is an individual-base model, i.e. it concentrates on the microscopic view to the system and at the same time it does not use *AI* techniques. To summarise, the major difficulties which arise when trying to develop a cooperative physical multi-agent system are the difficulties of implementing predefined path algorithms in a dynamic, changeable environment and within a large scaled system. In addition, the lack of interactive task allocation algorithms while agents are in motion is a consequent of the lack of sufficient communication. Finally, there is a need for suitable 3D visualization tools that allow the user to test, interact with the agents of the model, and modify the algorithm at interactive rates.

1.2 Main Contribution of the Thesis

This thesis introduces a novel interactive communication model for a $PMAS$. The communication model is described on two bases: the macro structure and the micro structure. The macro structure embeds the behaviour rules of the overall system; the set of interaction rules and the protocols via which these rules are executed. Hence, the communication model allows for agent negotiation and arbitration between two levels of communication; local and global communication. This implies that agents have a local communication to co-ordinate movements and a global communication for task level interactions. Regarding the movement co-ordination, a set of flocking behavioural rules is proposed to minimise the computations for the movements as agents only have a local view, local goals and knowledge, and resolves some of the conflicts that arises from the local interaction during the movement. In order to avoid the disorder results when agents no longer possess a global view of the world, a negotiation via a blackboard with a higher level agent is proposed for the purposes of the task assignment and performance. The negotiation via the blackboard allows all the agents to send/receive messages using speech act as a message passing medium.

The micro structure involves the architecture of the individual agents that meets the macro structure expectations. A Perceive-Think-Act (PTA) agent's architecture is introduced in the thesis. This architecture integrates the Belief-Desire-Intention (BDI) architecture and the blackboard architecture into a hybrid architecture that supports both: the reactive features of the blackboard and the deliberative features of the BDI architecture.

The thesis also introduced the use of virtual environments as a 3D-simulation and visualisation tool to test the communication protocol built for a $PMAS$. The 3D simulation for the model provides the user with a high level of interaction with the simulated world. Hence, the thesis presents the full implementation of the theoretical description of the communication methodologies in practice by building a scalable real time virtual environment VE. The proposed VE introduces a toolkit that allows an easy way to test and compare the different levels of agents' interaction under different architectures. This helps in studying the possible requirements for a hardware implementation of a $PMAS$ model.

1.3 Structure of the Thesis

The rest of this thesis is organised as follows:

- A theoretical background and the proposed MAS communication model is presented in chapter 2
- The formal model of the communication protocol is presented in chapter 3.
- Visualisation tools within the Virtual Environment Centre (VEC) are described in

chapter 4.

- Simulation Results are presented in chapter 5.
- How the proposed system would be implemented in real world, the hardware implementation, is presented in chapter 6.
- Discussion and Conclusions are presented in chapter 7

Chapter 2

Theoretical Background

Multi-agent systems (MAS) are becoming increasingly important in practical applications. Examples of these applications are data exchange presented in [72], and intelligent autonomous robots presented by Brooks in [17].

Despite the disparity of the areas in which these systems appear, interaction is a major feature and can be initialized between agents, whether they are similar agents (i.e. homogeneous MAS) or different agents (i.e. heterogeneous MAS). Lin *et al* [65] describes agents' interaction as a part of an extended communicative activities; e.g. dialogues, argument or negotiation among agents. Wooldridge states that defining the methods of interaction between agents is like scrutinising the communication protocols and negotiation processes of which agents are capable [107]. Specifying the negotiation technique in turn relies on the internal structure of the individual components, i.e. the agent's architecture, as well as on the proposed level of interaction between agents.

This chapter reviews the theoretical background of common communication and negotiation techniques, described in section 2.1. The common agents architecture is presented in section 2.2. Also, common visualisation techniques are described in section 2.3. Section 2.5 highlights the problems with the current communications and negotiation technique and the agent's architectures.

2.1 Common Communication Techniques within Multi-Agent Systems MASs

Common communication protocols use either message sending [38] or social behaviour-based interaction [30]. Meanwhile, negotiation processes by means of message passing is a main interest in the area of MAS. Although different views were presented by Cohen [28], Jennings *et al* [12] and Ferber [37], they agreed that negotiation by message passing plays a major role in the theories of communication.

Quintero *et al* [80] considered co-ordination and co-operation as the main stays of negotiation to be found in most applications of multi-agent systems. Co-operation and co-ordination are used for controlling and distributing tasks between agents as described by Kuwabara *et al* [57]. Accordingly, each agent needs to avoid goal conflict while it co-operates with other agents in a way that improves the overall system performance [12]. Cohen *et al* in [27] and [28] also stated that co-ordination increases the agent's capabilities to perform the specified task using the grouping technique. The grouping technique implies the agents co-operate by sharing tasks (team performance). Jennings *et al*[12] also identified conflict resolution by co-ordination in a MAS.

Quintero *et al* [80] describes common negotiation models and maps the solution to a given problem to the co-operative interaction between all agents in the system. Nwana *et al* [76] described a negotiation model that prevents duplicating the work and co-ordinates tasks. Jennings [12], Wooldridge *et al*[108] and [109] identified negotiation protocols as the set of rules that governs the interaction.

2.1.1 Communication via Negotiation

In AI, information exchange through blackboard negotiation is most often used as a group communication technique [37] for the purposes of co-operation and co-ordination. The blackboard is shared memory that is mostly used as a repository on which agents write messages, post partial results, and obtain information. Several applications employ this communication technique to achieve coordination [76] where agent's actions are interdependent and where such interdependent activities need to be coordinated. It prevents chaos, i.e. disorder, as agents do not individually possess a global view of the entire agency to which it belongs. Considering agents' interaction as a negotiation process by participating agents, they can now perform joint actions. Blackboard negotiation technique is the most common negotiation technique [76].

Current implementations of the blackboard architecture relies on the organisational structure; grouping and hierarchal structure. The hierarchal structure provides a way of ensuring coherent behaviour and resolving conflicts by providing the group with an agent which has a wider perspective of the system. This is the simplest coordination technique and yields a classic client/server architecture for task and resource allocation among slave agents by some master agent. The master controller can gather information from the agents of the group, create plans, and assign tasks to individual agents in order to ensure global coherence. Some such systems employ a blackboard architecture to achieve coordination, such as Werkman's DFI system [106] by distributed problem solving and the Sharp Multi-Agent Kernel (SMAK) system [54]. In this scheme, the blackboard's knowledge sources are replaced by agents who post to and read from the general blackboard. The master agent schedules the agent's reads/writes to/from the blackboard. Practically, Lesser and Corkill developed a Distributed Vehicle Monitoring Testbed system, described in [62], that exploits a blackboard architecture in which co-ordination occurs amongst peers as well as the user. Problem solvers via centralised co-ordination has also been used in markets for example in [98] which employs a global blackboard for posting buying and selling prices.

The above approach is impractical in many realistic applications because it is very difficult to create such a central controller that is informed of all agents' intentions and beliefs. Durfee *et al* [35] and [33] point out that such a centralised control that is presented in [41] or the above master/slave technique is contrary to the basic assumptions of distributed artificial intelligence. In addition, the current implementation of blackboard negotiation has shown that it allows for interdependent action co-ordination and not for motion co-ordination. Interdependent actions imply that an agent may need to wait on another agent to complete its task before executing its own.

Another implementation of the blackboard negotiation technique is to co-ordinate the global interaction between agents and with an external party. This can be carried out by allowing message passing through a blackboard between the participating agents and the third party [38]. The contents of these messages influences the agents' beliefs [31]. A blackboard negotiation implemented in this manner enables autonomous agents to influence others and convince them to act in a certain way. The contents of the exchanged messages are of main interest as they occupy a large area of research in the communication and negotiation in a MAS.

2.1.2 Communication via Speech Acts (SA)

Researchers in the field of multi-agent systems, e.g. Cohen and Levesque [26], focus mainly on the formal description of the contents of the communication acts not on the form. In this context, communication among agents in multi-agent systems has been fruitfully studied from the point of view of speech act theory. Speech act theory was first introduced by Austin [7] to deal with language utterances which were first identified by Searle [84] then [85] as the basis of communication mechanisms in multi-agent systems. Quintero *et al* [80] considered all utterances as actions executed by agents. SA has been used in designating the communication activities, i.e. conversations and negotiation by Cohen and Levesque [28] and Ferber [37].

A speech act SA is a basic communicative action that the agent can execute with no further consideration, with the aim of making some fact mutually believed or known by sender and listener. A speech act, SA, is an act that a speaker performs when making an utterance. A SA, is usually seen to have two parts: a conveying force (an illocutory force) and a propositional content. Each message includes one or more of the pair (conveying force and the proposition). The conveying force relates to the carrying out of the act by the sender on the addressee of the utterance; e.g. informing, asking to do, and ordering. The propositional content is the object of the conveying force.

Agents use speech acts to communicate their mental states. Hence, agents may use speech acts to exchange information about beliefs, goals, and intentions [14] via these units of communication. Beliefs-goals-intentions form the contents of the exchanged messages. As a consequence, a joint mental state for a team of agents may be established to enable the forming and disbanding of teams [90], and [31].

Speech acts have been classified into different kinds, e.g. declarative, expressive, promissive, assertive and informative SA [37]. A declarative SA gives the addressee facts about the sender's acts at the present. An expressive SA gives the addressee indication of the sender's mental state in the past. A promissive SA commits the sender to perform a certain act in the future. An informative SA plays a data-transfer role for information needed in a conversation to facilitate co-operation [37]. Informative SAs emphasises the donative function of the language. This implies that information are being donated by an agent to other agents without asking for a reply. An informative speech act takes place when the speaking agent tends to influence some mental state of a listening agent [37]. Consequently the listening agent changes its current mental state and, if required, performs a desired change in the environment [89]. Speech acts can fail depending on the listener's understanding of the message as well as the listener's willingness to perform the implicit directions within the received message.

Theoretical work in the field expects that informative speech acts are to replace the conventional speech acts which emphasises the conative functions of the language. Examples of speech acts that emphasise the conative function are expressive, promissive, assertive, declarative SA. In these examples the sender awaits the reply, e.g. questions are usually followed by answers, requests are usually followed by confirmations or refusals [26] and [89]. This was mainly used in knowledge query and manipulating language KQML [26] and Fipa agent Communication Languages ACL [38]. This SA that emphasis the conative function of the language results in:

1. Consuming time and resources.
2. Listeners still suffer some difficulties in understanding the received messages because of the ambiguity of the speech acts
3. SA can fail depending on the listeners willingness to perform the implicit directions within the received massage.

Considering a practical implementation of the informative speech acts enables the agent to send a sufficient amount of information; a feature usually called 'quantity'. An agent also will be able to only pass the right information according to its beliefs which is another feature referred to as 'quality'. This implies that an agent needs to pass information it believes is true [75]. This is an important issue when dealing with a group of agents that are assigned the same task and are asked to act together. They will be able to communicate their current beliefs and goals.

2.1.3 Communication via a Joint Action

Group communication, seen as co-operation, is also described in the light of the Joint Action concept. Cohen and Levesque [28] described agents' co-operation as agents acting together. This implies that a global behaviour emerges from the agents' individual actions as a consequence of the communication. Accordingly, the emergent global behaviour from simple local and individual behaviours has become an important factor in formalising the communication [31]. In this context, Cohen *et al* [27] incorporate the joint actions concept into the speech acts in order to form teams or groups to perform the joint task, discharge the joint tasks, form new teams and disband teams.

Panzarasa and Jennings in [77] believe that an advantage of linking the speech acts, described in section 2.1.2, to the joint actions concept is the differentiation between coordinated actions and joint actions. Tirassa [97] concluded that a joint mental state encompasses a joint intention of the agents in the same team. Accordingly, Kumar *et al* [56], Cohen and Levesque [24], and Levesque *et al* [63] defined a joint intention for a team of agents as an internal commitment to perform an action jointly demonstrating teamwork.

Communication via joint action, such as simulating a group of agents as a team in [99], has been shown to be efficient for co-ordinating a group of agents. However, current work on the agents' group communication theories has shown that they are still not capable of co-ordinating movements of a group of mobile agents, so, a motion co-ordination technique that controls the movements of a group of agents is still required.

2.1.4 Communication via Social Behaviours

Interaction between mobile agents in highly dense spaces and within real time is highly demanding. Co-ordinating the movements of such a group of mobile agents requires specifying an adaptable motion control technique. Examples of previously implemented methods, e.g. predefined paths and complex path planning algorithms [6], [53], when implemented in a large scale dynamic environment has been shown to be often computationally intensive and therefore unsuitable for real time operation. In contrast, communication via social behaviours has been studied and an example of an elementary technique used is the flocking algorithms, first introduced by Reynolds [29] as an adaptable motion control technique.

The flocking algorithm is used to simulate animal behaviour and related types of motion co-ordination within large groups. Within this technique, interaction-communication is governed by a set of common flocking rules described in detail in [29] and [104] and results in a system that automates agents motion and speeds up the reactions during motion. The agents' behaviour is fluid and gives a more realistic and natural behaviour when avoiding obstacles. In addition, agents movements according to this algorithm keep them naturally grouped but allows for splitting into teams as well as combining two teams into one team.

The earliest versions of flocking behaviour was simulated by creating a semi-predefined flight path for each agent [81]. This implied predefining a set of way-points for agents to move towards. Although many simulations developed are based on Reynolds system in which agents were modelled as point masses, a more realistic flocking model has been presented in [29], where the simulation of the model ran offline. Another example is presented in [104], that demonstrates a nonlinear aerodynamics model for actual Unmanned Air Vehicles and incorporated that presented in [29]. This model allowed the interactive control of the flocking parameters and investigated the relationship between rule weightings and the flocking behaviour. This model used homogeneous weights and rules for all agents.

Implementing the flocking algorithm minimizes the time required to perform co-ordination during motion [30]. During motion, flock members interact in real time within a dynamic environment and above all they do not need to have a prior knowledge about the environment. This results in an emergent group behaviour leading to emergent organisation.

Although these systems developed coherent flocking behaviour from simple rules and automate agents' motion, they lack:

1. Task distribution as a higher communication level.
2. Interactive visualisation where the user is able to monitor, assess and evaluate qualitatively the interaction between agents while they are in motion in a realistic simulated environment.
3. Scalability as these models simulate a limited number of agents with minimum representations and direction. The scalability is viewed on two bases: a) the real physical sizes of the simulated world and the moving robots inside this world, and b) the number of moving robots inside the world.
4. Locality in the computations which means the need to develop local controlled agents rather than global controlled agents.

5. Social Complexity that arises when moving the scale of the current flocking systems from small groups to large groups. This is because all agents within the system are related to each other and they all communicate. Accordingly, there will be an increase of interaction overhead [103]. This denotes the increase in the communication between agents required to detect an interaction and coordinate their activities [91].

6. Quantitative investigation of the interaction between agents including the distributed weights and rules where each agent runs a suitable rule and sets the corresponding weight to the chosen rule.

Considering both social complexity and the corresponding increase of interaction overhead leads to defining the level of locality and organising agents into groups so that they do not all communicate. This raises two main questions. First, to what extent does this agent need to know about the surrounding area and the nearby agents? Second, with whom do these agents need to communicate? The answer to the first question requires defining the range within which an agent perceives the environments. In addition, it is important to define both the fields of view for each agent and the social relations, i.e. the grouping of the agents into teams.

2.2 Common Agent's Architectures

This section describes what forms an agent architecture and some of the common architectures avilable. There are three key issues needed to build an agent's architecture. These are: the properties, capabilities, and the type of environment.

Agent's properties include how this agent is related to the organisation and how it is controlled. This implies defining the relations between agents inside the world and also defining the relations between agents and the other objects in the environment.

Capabilities include an agent abilities to act and learn. In order for an agent to be able to interact, communicate and cooperate within an organization it belongs to, it needs to be: a) adaptable: this implies that an agent has the ability to respond to an event in a dynamic environment within an acceptable response time, b) versatile: this implies that an agent has the ability to vary its responses dependent on what beliefs it has and on the current environmental status, c) real time response is essential as an agent has the ability to sense the environment and create an adequate response to it [108]. An agent that possesses these capabilities can move and interact autonomously with the objects inside the world.

The above capabilities require an autonomous architecture that is capable of interacting with its environment via its perception modalities (described in section 2.4.2) in order to move and accomplish some task. At this stage, it is essential to decide whether it is a reactive architecture that is required or a deliberative one?

2.2.1 Hybrid Belief-Desire-Intention Architecture

Arguments against both purely reactive and purely deliberative agent architectures are presented in [49]. An architecture that satisfies both features is the Hybrid Belief-Desire-Intention BDI architecture [48], [108]. The BDI-architecture defines five main components of the agent architecture: sensor, actuator, communication, cognition, and intentions. In the hybrid BDI architecture, agent control subsystems are arranged to deal with information in a hierarchy. For example, sensor data might be dealt with directly, while the higher-most layer deals with the long-term goals referred to as intentions. Wooldridge and Jennings [108] addressed a key problem with the conventional hybrid BDI architectures which is the lack of a control framework to manage the interaction between the subsystems [108].

2.2.2 Blackboard Architecture

Another agent architecture that serves well in co-ordinating multi-agents is the blackboard architecture [79] and [32]. The blackboard architecture is built as follows: a) it has one input channel which delivers new data caught from the outside world by a perception system to the agents internal subsystems as sensory data or received messages. b) it includes an interpreter that has three main functions: it updates an agent's beliefs from the information caught by the perception system and generates a new set of desired actions on the basis of the new beliefs. Then, it decides the desired action as an intention. The selected desires are a result of a decision making algorithm that prioritises and determines the degree of importance of the desires via a set of weights. c) As a result, a single output is executed through the output channel, also known as the action system, to influence the outside world.

A blackboard architecture gives the agent distributed capabilities so that it can deal with a variety of events at the same time. This is because it allows all subsystems to operate asynchronously via the interpreter. The blackboard architecture also gives explicit structure to the agents knowledge that allows an agent to know all relevant information about its domain. In this case, learning is not required for understanding the events in the domain. This implies that the behavior of the system is dependent on perceptions [79]; i.e. it supports a perceive-act agent's architecture. This is especially useful for building physical agents [59].

The blackboard architecture is based on the concept of having a pool of knowledge through which different agent activities can communicate [43]. Therefore, the blackboard architecture, as a multi-agent organisation, allows exchanged messages to be passed through to all other agents in the system, see section 2.1.1.

2.3 Common Simulation and Visualisation Tools

Simulation tools enable us to sufficiently understand the way these agents move and interact inside their world by visualising their behaviours. Hence, simulation can highlight the potential problems at an early stage, allowing for a quick and easy modification.

2.3.1 Why 3D simulations

The ever increasing evolution in computer graphical technologies helped in the rapid change towards using 3D simulations in testing theoretical models. This is important because the lack of the realism factor in 2D simulation leads to the production of an incomplete view of the simulated world. Realism is the main feature in 3D simulations. Realism can be defined as geometrical realism and behavioural realism. The former describes to what extent the simulated world has a close appearance to the representation of the real world

[88]. The more realistic, an object's representation, the more realistic views the user gets. The latter implies the existence of behavioural signs that indicate interactive responses which cannot be caused by geometric realism but because of the realistic responses and behaviour. Realism is considered to be important in order to grasp and get a reasonable sense of dimensions, spaces, and interactions in the simulated world. Therefore, for more realistic representations for agents and the virtual world as well as their behaviours in the simulated world, the 3D-simulation and visualisation tools are of great importance. Desktop simulations can not afford this level of realism in a simulation.

In order to assess and test the agents different behavioural level and modify the algorithm in real time and at interactive rates, the user needs to, for example, interactively monitor, test, and modify agents' behaviours during the development process. Current common visualisation techniques used to simulate behaviours of communicating agents rely on simulating agents movements and their world often displayed on a 2D desktop.

2.3.2 A Comparison between 2D and 3D-Simulations

3D simulations can be either desktop or immersive simulations. In spite of the fact that both types of 3D simulation satisfy the realism requirements to different levels, displaying the 3D simulation in a desktop size cannot be considered sufficient when simulating a large number of agents moving inside a simulated world with real physical spaces and sizes. A 3D immersive simulation can be more efficient for visualizing all the agents at once. 3D immersive simulation is important to improve the impression of presence. Presence has been simply defined as the sense of being there within a Virtual Environment by filling the user's field of view and hearing by the information from the virtual world. Recently, Slater [88] stated that presence is not only the feeling of being there, but, also the existence of behavioural signs is essential to obtain the impression of being there, in other words these signs defined the concept of behavioural presence.

Behavioural presence is the concept we are interested in in this work. It is especially useful to test new situations, to initiate new goals or to change the original goal and monitor the system performance in real time. As a consequence the user can gain more understanding of the behaviour and simulate more believable and realistic behaviour.

Another point that is considered important within the immersive 3D simulations is the real-time walkthrough feature. A real-time walkthrough allows for human intervention during the system run, moving closer to the objects inside the simulated world and interactively initiating new events, communicating with objects, and monitoring closely the resulting reactions. This human intervention can not be afforded using 2D simulations.

In summary, building a virtual reality system that represents multi-virtual agents that co-ordinate with each other produces more convincing simulations than the conventional desktop models could offer. This improves the ability to simulate the believable behaviours because VE supports three main features: realism, presence, and immersion. The level of realism, presence, and immersion indicate the quality of the produced 3D virtual environment. Accordingly, the user is able to interact efficiently with the simulated world, as well as monitor, and visualise the emergent behaviour in real time.

2.3.3 Current implementation of VEs

3D simulation not only need to mimic the real physical spaces and but also needs to support real time visualisation and monitoring [94]. This is essential for a large scale interactive virtual environment application. A recent example of an interactive virtual environment system was presented in [21] where a virtual museum's assistant takes the tourists in a virtual tour and interactively answers questions.

Other examples of current utilisation of full scale 3D semi-immersive simulations include simulating an urban space environment, an interior architectural environment, and simulating the behavioural models for the training purposes. Example of recent 3D immersive environments that have been used successfully to develop group behaviour models, and perception models are: simulating the movements of crowds [92], simulating a fire evacuation system [64], intelligent transportation systems [101], urban behaviour development [93], and an intelligent vehicle model for 3D visual traffic simulation [102]. These examples have shown that the immersive 3D simulation VE supports training purposes as they feature: presence, realism, and immersion which all add value to the simulation.

Virtual Prototypes and product evaluation is another application that exploits virtual environments technologies. It propagates the idea of a computer based 3D simulations of systems with a degree of functional realism. Accordingly, the virtual prototypes are used for testing and evaluation specific characteristics of the product design. This can guide the product design from idea to prototype which helps to address the engineering design concerns of the developer, the process concerns of the manufacturer, the concerns of the maintainer, and the training and programmatic concerns of the operation. Simulations of these systems are being developed to enable the creation of a variety of realistic operational environments. Virtual prototypes can be tested in this simulated operational environment and during all the development stages. Once a virtual prototype is approved, design and manufacturing tradeoffs can be conducted on the virtual prototype to enhance productivity and reduce the time required to develop a physical prototype o even direct to the final product.

Another useful area where real time semi-immersive simulations can be useful is controlling the movements of mobile physical robots, and assessing the individual behaviour in a robotic application [74]. An example is presented in [61] which provides a working model of its autonomous environmental sensor for telepresence where the robot tours the inside of a building and automatically creates a 3-D map of the interior space. An example that presents how virtual reality helps in handling the connection of the various tools and the communication between the software and the hardware of the robot is presented in [82]. Simulating such systems require real time visualisation, a high level of geometrical realism in representing the world and the moving objects, and support for the specifics of the application. For example, in order for a set of agents, representing a set of autonomous mobile robots, to build their knowledge system they need to perceive the environment. Accordingly, specifying a sensing device and sensing techniques is essential. This can not be achieved by implementing point mass objects. Alternatively, the computations need to be dependent on the sensing devices and rely on real data gathered from the world.

To summarise, the use of large scale visualisation tools in recent published work included the areas of assisting architectural design, urban planning, safety assessment, and virtual prototyping. Using a large-scale virtual environment in evaluating robotic products and applications has been used for simulating and testing the hardware whilst producing single-robot applications. Using a 3D full scale semi-immersive environment for simulating communication and co-operation between multiple robots has not yet been fully covered. Major anticipations of using 3D-simulations are:

- 3D immersive simulations can support realism in representing the simulated world in order to sense the dimensions and spaces, these terms are referred to as *geometrical realism*.
- 3D immersive simulations can support the specifics of the simulations for example simulating the *sensors*.
- Allows the user interaction with the simulated world at interactive rates, referred to as *behavioural presence*.
- 3D immersive simulations must allow for real time visualisation of a large number of agents displayed simultaneously in real time, referred to as *scalability*.

2.4 Hardware Implementation

Co-operation within a multiple mobile robots system requires these robots to communicate. In the real world, communication implies that those robots possess both communication mechanisms and tools. The communication tools presents the way and the devices that connects these robots to the world. The connection devices, for example, can be cable or wireless. Most of the $MMRS$s' applications require wireless connection via infrared devices, sonar sensors, or wireless network connection to support flexible movements. The communication mechanisms starts from the inputs to the agents' internal systems via one or more of these communication devices. These inputs represent the base on which the whole interactions are built for performing the tasks. A very basic task that is based on the robot's interaction with the world is navigation. Building mobile agents that attempt to navigate and act in a dynamic environment must have efficient navigation and perception strategies.

2.4.1 Navigation

Autonomous navigation represents a higher level of performance, since it applies obstacle avoidance simultaneously with robot steering towards a given target. Therefore, real time obstacle avoidance is one of the key issues to a successful application of any mobile robot systems.

Current navigation algorithms in the area of multiple robot systems use either an artificial potential field approach [16], optimal path algorithm [15], or flocking algorithm, described in details in section 2.1.4. The idea behind the potential field is to: a) define a space which is a free for a robot to move in, and b) this space is determined by a number of attractive and repulsive poles. The effect of the attractive pole is to cause the robot to move towards them. The regions in the space that are defined as obstacles are modelled as a repulsive force or high potential, whilst the destination to which an agent is required to move is modelled as a low potential. Using the potential field algorithm enables continuous motion of the robot without stopping in front of an obstacle. This requires not only detecting obstacles but also requires some kind of real time quantitative measurements. A collision free path is then derived by following the line of highest attractive potential from the start point to the destination point. The major problem with the potential field approach is that it is subject to local minima. Since an agent tends towards lower potential areas, it can reach a state of equilibrium, and accordingly become trapped. Another issue can lead to trapping an agent somewhere that is a goal-conflict trap; for example an agent might encounter an obstacle whilst moving towards a target that lies behind that obstacle.

These trap situations can be solved by considering suitable adaptable techniques that allow an agent to switch between different modes of motion and communication techniques to suite the case. In addition, current research has shown that distance computations and obstacle avoidance are in great demand; this is because each robot needs to find out the positions and distances according to its local co-ordinate system. Adaptable motion control techniques allow the agent's priorities and actions to be updated via a set of weights, e.g. flocking algorithms.

Mataric [70] and [71], produced a dual perception flocking behaviour pack of real robots. According to Mataric's pack, robots use infrared for obstacle avoidance and sonar for collision avoidance. Ferber [37] described Mataric's pack as difficult to implement for real robots due to the limitations of the sensors. Brook [17] put forward an elementary and very simple capability animal-like intelligent navigation technique embedded in inexpensive small robots. These are purely reactive robots. Bererton *et al* described similar reactive robots in [13]. Recently, a set of seven flocking dwarf robots was introduced in [11]. This model also uses the dual perception system for the flocking behaviour as the robots used infrared for intercommunications whilst they used the sonar for obstacle avoidance. Yet those flocking robots are not able to communicate with a higher level agent for the purpose of interactive task assignment.

These projects have shown that interactive communications with a co-ordinator or a user has not been fully considered. These flocking robots are not able to communicate with a higher level agent for the purpose of interactive task assignment. We need for the proposed solution a set of deliberative robots that can interactively, perceive, think, and then act. They will only implement an animal intelligence technique which exploits only reactive architectures for the physical agents. In contrast, Langley *et al* described theoretically the cognitive structure of knowledge in [59], which has no links to the communication model.

2.4.2 Perception

Perception is considered as another fundamental robotic task and refers to the extraction of knowledge from the environment. One characteristic of perception is that it may integrate information from different modalities. For example, in humans the modalities of perception correspond to the five senses: taste, touch, sight, sound, and smell. In the context of an agent's communication, different perception modalities have been identified for the different communication activities. Common modalities of perception are sensors, and internet connection. Although, sensors are considered the main perception modality, there are a number of problems caused by the common sensing strategies. These are:

1. Environmental complexity and the agents computational capabilities

   Both real and simulated dynamic environments can be very complex. Complexity in this case includes both the enormous amount of information that the environment contains and the enormous amount of input the environment can send to an agent. Accordingly, the amount of perceptional information to be processed is often greater than the computational capability of the agent. As a result, the agent must have a way of managing this complexity in order to respond sensibly to relevant information in real time. If we consider a small robot of radius $30cm$, using a larger processing units onboard will be considered impractical.

   Solving the information overload requires using efficient strategies that allows filtering out parts of the perceptual field and paying particular attention to others; this is known as an attentional mechanism. Another mechanism which is usually useful, is to filter out the information of an irrelevant type; i.e. focus attention on relevant percepts. In this case, the agent makes a deliberate decision to concentrate on particular environmental percepts and must be forced (perhaps by a high weight) to prioritize its desires and move its attention elsewhere.

2. Incomplete Information or Delay in Sensing

   This section discusses the opposite situation to the information overload described above. Sometimes the sensors provide incomplete information and the state of the agent is always behind the state of the external environment; this is known as delay of sensing. At other times the perception also is corrupted by faulty readings or some other problem when accurately sensing the environment. In this case an agent loses the connection with others in the system. The agent's internal system needs to have a failure tolerance recovery that protects an agent from reaching this situation. The solution can be the implementation of a bimodal communication system. This is based on both informative messages, for the purposes of task assignment, together with a motion co-ordination method for the purpose of controlling the motion. Accordingly, each agent can arbitrate between the two modes of communication. As a result, when an agent loses the connection with others in the system it can still move and search for other team members. Alternatively, it can still reach the target even if it loses the connection with the team members as it possesses the same joint intention whilst committed to perform a joint action.

To conclude, the common communication system for intelligent agents considers the theories of agent's architecture focused on purely intellectual tasks, i.e. software agents, and does not fully covered issues of perception and action in the physical world [58]. Therefore, the ability to transfer the theories of communications within a real world implementation requires studying the real world tasks and constraints. Navigation and perception tasks are fundamental tasks in robotic application.

2.5 Summary

This chapter has presented a review of common communication, and visualisation techniques. In addition it has presented the common agents' architectures. Regarding the common communication techniques presented in section 2.1, it was found that:

- The current communication techniques between large groups of agents suffers from the social complexity that leads to include all the agents in the system in the computations, described as interaction overhead described in section 2.4.2.
- Common communication by social behaviour systems implements the flocking algorithm as a motion control technique without assigning any higher task.
- Communication via a blackboard negotiation technique supports action co-ordination in the sense of action collaboration but not action cooperation. This implies that it does not support motion co-ordination for a large group of moving agents.
- The common communication techniques lack any interactive communication within a hierarchal order that allows an agent to communicate with other agents and the user in real time.

The communication between agents requires the consideration of an appropriate agent's architecture. Section 2.2 showed that deliberative architectures such as hybrid Belief-Desire-Intention BDI lacks the framework that manages the interaction between the outputs of the agent's subsystems, whilst the blackboard BB architecture suffers from insufficient input channels that support an agent with different levels of information.

In order to visualize and assess the agents' behaviours that emerge from the interactions and communications, it is important to review common visualisation and simulation tools, section 2.3. It has been found that simulating real physical spaces and sizes, moving around the objects inside the world and interacting with them are not truly available within desktop simulations. Also, desktop simulations do not support complete interaction and navigation inside the simulated world. Hence, this raised the need to produce full scale real time simulations that presents things as they really are, and immerse the user in what he sees. Another contrasting point is that desktop simulations may have limited capabilities of zooming in and out for local and global views. In addition, desktop simulations can not display a large number of agents at realistic sizes. In contrast, large scale 3D immersive simulations can help the user to qualitatively test the communication algorithm and helps to test whether it simulates the expected behaviour.

It is therefore identified that in order to develop a framework for a communication system, within a large scaleable real time co-operative multi-agent system, the primary requirements of such a framework can be listed as follows:

- Defining social relations between agents by grouping the agents into teams which reduce the communication overhead. Accordingly, we can analyse the system behaviour under different team sizes, to aid defining a suitable team size and number of teams.
- Specifying two different perception modalities and the two input channels, that enable an agent to acquire the knowledge about the world on both levels, locally and glaobally. This implies specifying sensors, message passing techniques, and the form of the messages to be exchanged between agents that feature both the quality and quantity, see section 2.1.2.
- Specifying an efficient way that allows an agent to manage the acquired information.
- Utilising efficient simulation and visualisation tools that allow the user to monitor, assess and interact with the agents in real time.

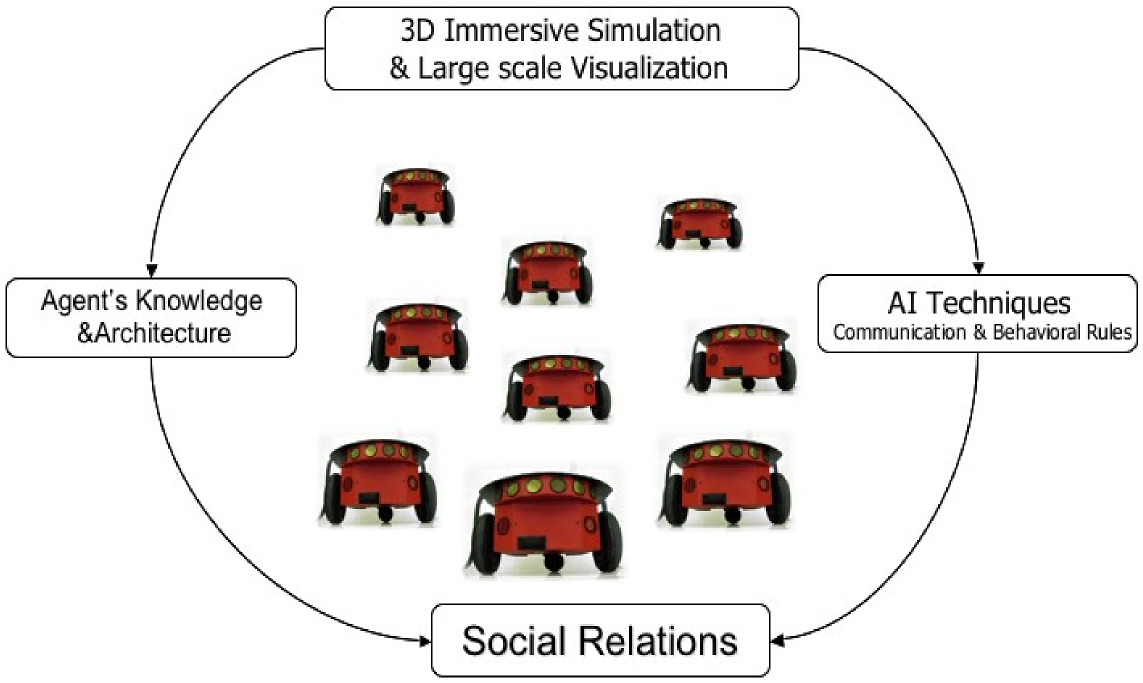


Figure 2.1: High level abstract system structure

The work presented in this thesis uses a multi-agent approach that aims at developing a communicational system that meets these requirements.

Chapter 3

Computational Framework

The behaviour of a multiagent system (MAS) can be defined by how its agents interact. In a MAS, each agent does not only need to be able to do the tasks that arise locally, but also needs to interact effectively with other agents.

An important part of the interaction between agents takes the form of communication. A good interaction implies efficient communications among agents, and therefore implies good performance. A protocol is the specification of these interactions/communications. This chapter is the core part of the thesis as it discusses the theory of the communication developed across two levels in a bottom-up model, these levels are:

- The micro structure of the MAS: this relates to the design and the structure of internal components of an agent. The micro level part is designed to provide the empirical specifications of a physical agent's knowledge and architecture.
- The macro structure of the MAS: this relates to the design and construction of

the agents' society that includes the specification of knowledge-rule based interaction techniques between the agents.

This chapter is organised as follows: the micro-level structure is presented in section 3.1 including: an agent's architecture, section 3.1.1, the proposed sensing strategies in section 3.2.2 and an agent's information management, section 3.2.1. The macro level structure of the system is presented in section 3.3 including a full description and implementation of: the flocking algorithm as a motion control technique at a local level, section 3.3.1, and the novel utilisation of the blackboard negotiation with speech acts as a global communication mechanism, section 3.3.2.

In order to put the communication rules, described in sections 3.3.1 and 3.3.2, together, it is essential to look at how MASs are modelled. If the actions of the agents are described as movements, the MAS model must be a physical and geometrical model, (see section 3.4). Physical MAS models are particularly useful for MASs that are composed of little robots moving in an environment and at the same time communicating with a higher layer agent; e.g. user who can be considered as a GBC. In this respect, agent's acts are classified into two categories: physical acts and communicative acts. The agent's physical acts are being determined according to the computational basis, section 3.4. On the other hand, an agent's communication with the GBC is modelled in terms of communicative acts being sent to the GBC, section 3.5, as it formalises agents' messages and presents the proposed communication modes explained with some examples.

3.1 Micro Structure of the MAS

Based on a cognitive physical agent design, that operates in a real mobile multiple robot system, a perceive-think-act architecture (described in the next section) allows for knowledge from different sources to be processed simultaneously. The way an agent manages the input information from different sources is described in section 3.2.1. Finally, information delivered by the sensors, as one of the input devices, is controlled by some of the sensing strategies, described in section 3.2.2.

3.1.1 The Agent's Architecture

In order for an agent to to be able to interact with other agents in the environment according to the previously described co-ordination concepts, a perceive-think-act agent's architecture is required. A perceive-think-act architecture that supports a perception dependent behaviour is considered; this is the Hybrid Belief-Desire-Intention ($HBDI$) architecture, described in section 2.2.1. In the $HBDI$, each subsystem (rule) continuously produces suggestions that the agent should do as the next action, and is combined with a blackboard architecture (BB) that allows an agent to arbitrate between the different outputs of these subsystems.The blackboard architecture organises and manages the different suggestions for the interaction between the subsystems outputs. A key enhancement is proposed to this architecture; a metalevel component. The basic idea behind this component is to run a world model faster than real time to make predictions of future states. These predictions are used to guide the agents behaviour at an advanced level. According to this

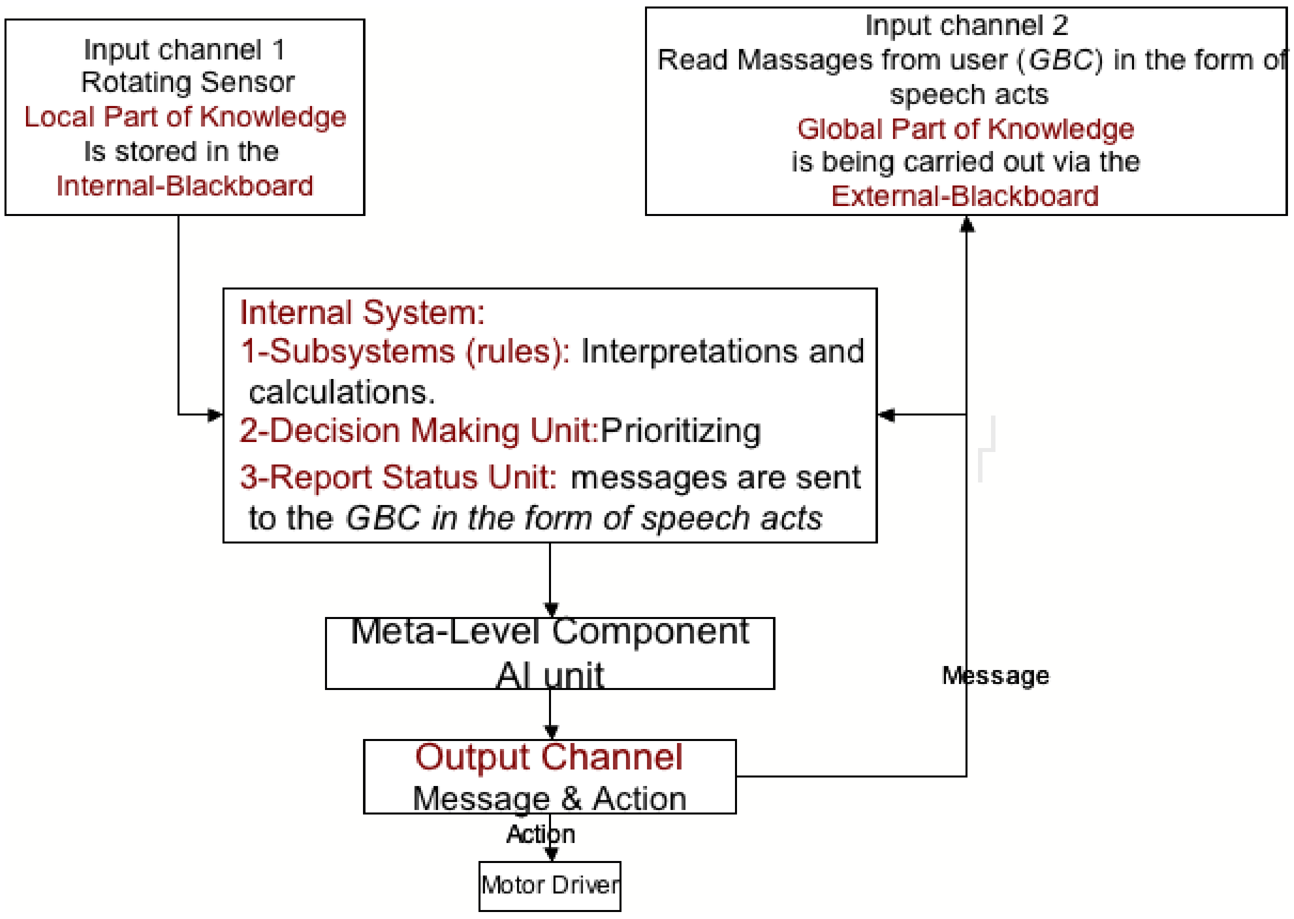


Figure 3.1: The agent's architecture

approach, the state space is divided into desired and undesired states. When the meta-level component detects that the system has reached an undesired state, it modifies the selected action in order to try and avoid reaching this state.

To summarize, for the practical implementation, the $HBDI$ architecture and the BB architecture are integrated into one new architecture, a Hybrid Belief-Desire-Intention-BlackBoard architecture ($HBDI - BB$) to suit the physical agents applications. The next step is to understand how an agent conceives and then manages the input information.

3.2 An Agents Knowledge Representational System

The capabilities of an intelligent agent is limited by its knowledge, and its computing resources. In order for an agent to understand the outside world and to interact with it, an agent's knowledge representational system (KRS) is required [37].

An agents knowledge can be described as the preconditions of actions [97]. For example, an agent believes that an obstacle exists in the way and it decides to turn right in order to avoid this obstacle. This agent's belief, i.e. knowledge, was *encountered an obstacle* and the agent's intention, is to avoid the obstacle. The next action, was *turn right* to avoid it. The agent's belief and intention are the preconditions of the action 'turn right'.

The new set of beliefs and intentions for an agent structures its mental state. At any moment, a mental state comprises a set of cognitive units: intentions and beliefs [37]. Representing these cognitive units and interpreting them using a set of words or temporal strings, structures an agent's KRS [52]. An agents KRS constitutes what an agent needs to know in order to be able to perform the next action, see figure 3.2.

Cognitive units are classified into two categories. First, factual cognitive units that describe the agent's current state, including the position, distance from other perceived agents and speed. Second, provisional cognitive units that describe (possible) intentions that an agent could have depending on its beliefs [37].

An agent's mental state results from the interaction between its cognitive units [77], in other words each elementary cognitive unit is linked to others' cognitive units to form a dynamic mental configuration that changes with time. This implies that any provisional

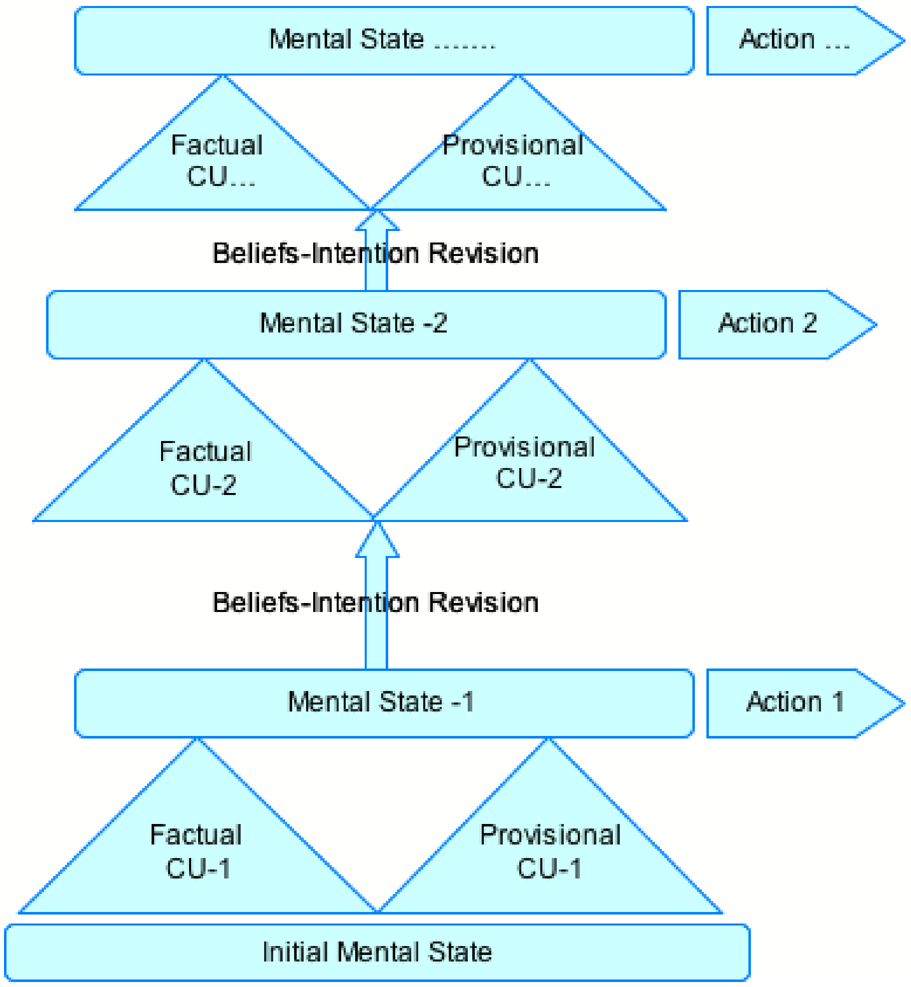


Figure 3.2: The agent's mind

cognitive unit can be created or can disappear as a consequence of the creation or disappearance of other factual cognitive unit(s) in the KRS. For example, when an agent encounters an obstacle, a factual cognitive unit that represents part of the agents overall mental state requires the agent to avoid this obstacle. But as soon as this agent avoids the obstacle, this belief is no longer true. As a consequence of changes in the environment (e.g. disappearance of this obstacle), an agent has to revise its beliefs on the basis of any new information it receives, together with the knowledge it already possesses. This belief revision leads to a dynamic mental configuration as a result of cognitive dynamics in the light of changes in the environment [97]. Accordingly, an agent's cognitive dynamics must be tightly coupled to the changes in the environment, this implies that they depend on the adaptive interaction between its cognitive system and the surrounding (mental, bodily, physical, and social) environment.

In the case when two opposite intentions are present at the same time and these need an agent to act in a joint and opposed manner, the agent then arbitrates to which of its states it will respond depending on the cognitive components it has. Accordingly, an agent ignores some of the KRS components such as beliefs-desires-intentions. Therefore, an alternative knowledge management that can be thought of as an interaction between all the present targets, may aim at a new target that averages these targets to define its next action.

3.2.1 An Agents Sources of Knowledge and Information Management

The proposed agent's architecture implies that an agent needs an artificial perceptual system that allows it to gain some knowledge about the surrounding world. It also implies that this architecture allows an agent to manage the various inputs to produce one single output as a desired action. The desired action is produced on the basis of active bi-modal perception; i.e. sensing and message exchanging. Each agent is provided with a sensor, through which it gains the local part of its knowledge and the global part of its knowledge is gained by reading the global-blackboard messages. All the information, the sensory data and messages, are represented and stored in the agents KRS, see table 3.1.

Table 3.1: An agent's information management and sources of knowledge.

Manager	Source of information
Internal-blackboard	-An agent gains part of his knowledge about the local environment via its sensors, - Local Communication.
External-blackboard	-Agents read/write messages from/to *GBC* via two ports 1- Input port which carries out the sent massages from agents to the *GBC*. 2- Output port which carries out the massages from *GBC* to teams.

Recall that an agent's mental state represents the interaction between a number of cognitive units (section 3.2), each agent starts from its initial mental state, then interactively acquires knowledge about the surrounding world, seeking new information, figure 3.2. Sensors are used to gain the local part of the information using a suitable sensor range. The local input information to the agents internal system is organized within an internal-blackboard; e.g. the identities and the positions of the detected agents. This part of the blackboard, the internal-blackboard, involves all the information required for local interaction with near agents.

Messages are exchanged via the external blackboard and they are used to gain a global view of the world. These messages involve the long term actions to be performed. The global information about the outside world such as allocated tasks are passed from a Ground Based Controller (*GBC*) to the agent via the external-blackboard as an indirect communication process. Here, the *GBC* represents a second layer that distributes tasks amongst teams. The global communication process via the external-blackboard is considered as a negotiation process between agents and a higher level agent (the *GBC*), although all agents can read all messages but an agent only sends messages to the *GBC*. The mechanism in our system for exchanging messages is based on the 'speech act' as a communication utterance. Accordingly, an agents knowledge about the changes in the environment results in a message that is classified as an intentional communication message.

An agent thinks and produces a desire (present-directed-intention) taking into account the long-term goal (future-directed intention). Accordingly, any new behaviour is then produced as a result of the change of an agents mental state.

3.2.2 The Agents Sensor and Sensing Strategies

To build a system based on a physical grounding hypothesis, it is necessary to connect it to the world via a sensor and actuators. The sensor returns identities and distances to the nearest detected agents. For each time interval τ, the sensor data is stored in one location of an (1×20) array as a part of the agents knowledge of the world.

According to the sensing problems mentioned in section 2.4.2, the information stored here is filtered whilst interpreting it by each rule via two main sensing strategies. First, the irrelative information is filtered out. In this case, the agent makes a deliberate decision to concentrate on particular issues in the environment. For example, an agent needs only to consider the user and other agents in the same team for the interaction rules, whilst it needs to consider all the objects for the collision avoidance procedure.

Another useful mechanism is to filter out parts of the perceptual field and pay particular attention to other parts; this is known as an attentional mechanism. For this, the caught information is filtered according to the requirements of each rule, e.g. an agent need not avoid colliding with an agent located behind it. The knowledge-based interaction/communication rules require that the perceptional field defined for each rule varies depending on the aim of the rule and the degree of importance of the information needed for that rule.

3.3 Macro Structure of the MAS

In this work, the theory of communication is built on exploiting: a) an animal intelligence technique, described in section 2.1.4 to co-ordinate movements, and b) cognitive intelligence technique, described in sections 2.1.3 and 2.1.1, to co-ordinate actions in order to avoid conflicts.

The key movement co-ordination exploits a flocking algorithm to mimic the motion of a large group of animals and the action co-ordination concept exploits the blackboard negotiation technique, i.e. communication is considered as a negotiation by passing messages from the GBC to agents for task allocation, and from agents to GBC in order to report status and findings. All tasks are distributed within a hierarchal order via a ground based controller– a user. Speech acts are used to compose the exchanged messages via the blackboard.

Accordingly, the proposal identified that two levels of communication are required: local and global. The local communication deals with the information that is delivered by the agent's sensor to co-ordinate movements. The global communication deals with the hierarchal task distribution with the user, a ground based controller (GBC). Subsections

3.3.1 and 3.3.2 presents in details the rules of interaction/communication.

3.3.1 Local Interaction Technique: Flocking Algorithm

This level of communication controls the interactions between local agents. The local communication is considered as an agent-to-agent negotiation using flocking simulations to coordinate movements. It holds the lower-level information, this includes the identities of the sender and recipient, positions, and orientation. Hence, it considers the agents sensors as the only source of its knowledge.

The flocking behaviour can be achieved by applying three rules: alignment, cohesion, and collision avoidance. Each rule, also considered as a subsystem, produces a suggestion for the interaction. These suggestions are simply: a centroid[1] where an agent needs to head towards, a correction angle an agent needs to modify its direction, and finally a weighting value to determine the importance of this rule.

Three enhancements are proposed in order to address the list of problems with the conventional implementation of these rules, discussed in section 2.1.4. First, a local perception zone is used (that imitates the perception zone an animal may have in reality) rather than the conventional global one. This results in localising all computations so that only local agents are considered. The range of an agent's sensor controls the level of locality for an agent so that it can only interact with nearby agents. Therefore, it is essential to specify the sensor range and sensing strategies in order to meet the locality and real time requirements.

[1]A centroid, in some cases, is an average spatial position for the positions of a set of agents. The centroid also can be the distance to a single agent, in other cases.

The parameters for the perception zones are illustrated in table 3.2. The values of these parameters are then specified with the aid of the visual simulation in order to ensure a suitable level of locality and to support the role of each rule.

Table 3.2: Perception zone for the three flocking rules

Perception Zone	Alignment Rule	Cohesion Rule	Collision Avoidance Rule
S_d	10 *units*	10 *units*	7 *units*
FOV	360 *degrees*	360 *degrees*	36 *degrees*

The second enhancement is in the way an agent prioritises its actions. Within the previous systems, the final decision is often made by giving the priority to the collision avoidance rule, i.e. if any object appears in the perception field, all the weights for the other rules will be reset until the agent avoids the object. According to the proposed new protocol, all the weights are considered regardless of the existence of any obstacle. The third enhancement is the use of dynamically computed and distributed weights and rules. The conventional implementation of the flocking systems specifies homogeneous rules and weights in a way that all the agents use the same rules with the same weights. This results in global centroids and orientation. According to the above enhancements, the proposed new three flocking rules are described as follows:

- **Alignment Rule** R_α This rule is also known as a velocity matching rule as each agent A_i searches for the nearest neighbour from the same team A_j and tries to align its velocity vector with the velocity vector of this neighbour. The sensory data is filtered for this rule to pick only the nearest friend, a detected neighbour from the same team. The nearest neighbour is another agent that falls within a 360 degrees field of view and exists within a range equal to the agents sensor range. The sensor range and the field of view defines the perception zone for this rule.

The alignment rule results in a velocity vector an agent should follow to modify its current direction in order to align with this neighbour, if found. For agent A_i, this vector composes a centroid $C_{A_i}^{R_\alpha}$ and a correction angle $\theta_{A_i}^{R_\alpha}$. The centroid of this rule is considered as the distance to agent A_j positioned at P_{A_j}, (see equation 3.1). The correction angle is computed as the difference between the current heading Ψ_{A_i} of agent A_i and the heading angle α_{A_j} of the nearest friend A_j, (see equation 3.2).

$$C_{A_i}^{R_\alpha} = P_{A_j} - P_{A_i} \tag{3.1}$$

$$\theta_{A_i}^{R_\alpha} = \Psi_{A_i} - \alpha_{A_j} \tag{3.2}$$

This rule is implemented by the common flocking systems and this proposal adopts the same rule but with different weighting strategy. The weighting value $w_{A_i}^{R_\alpha}$ is dynamically updated and is computed as a reciprocal of the centroid magnitude. The weighting value is used to adjust the strength of the alignment force. The outputs from this rule $< C_{A_i}^{R_\alpha}, \theta_{A_i}^{R_\alpha}, w_{A_i}^{R_\alpha} >$ are stored in the internal blackboard and used to partially modify the agents current velocity.

- **Cohesion Rule** R_β

The cohesion rule acts as a binding force. It reinforces each agent to orient its velocity vector towards the centroid of the team members that fall in the field of view of this rule (360 degrees) and exist within a range equal to the agents sensor range. For an agent A_i, located at corresponding positions P_{A_i}, the centroid $C_{A_i}^{R_\beta}$ of this rule is computed as the distance to the average of the positions of the m detected A_js located at position P_{A_j}s. The agent A_i computes the distance to the centroid $C_{A_i}^{R_\beta}$ and a correction angle θ^{R_β}.

$$C_{A_i}^{R_\beta} = \frac{1}{m}\sum_{j=1}^{m}(P_{A_i} - P_{A_i}) \tag{3.3}$$

$$\theta_{A_i}^{R_\beta} = \Psi_{A_i} - \beta_{A_j} \tag{3.4}$$

Similar to the alignment rule, a weighting value $w_{A_i}^{R_\beta}$ is used to adjust the strength of the cohesive force. The weight value was empirically set according to the experiments described in section 5.2.4. The outputs from this rule $< C_{A_i}^{R_\beta}, \theta_{A_i}^{R_\beta}, w_{A_i}^{R_\beta} >$ are also stored in the internal blackboard.

- **Collision Avoidance Rule R_γ**

This rule prevents an agent from colliding with other objects and agents in the environment. It also helps avoid overcrowding. An agent uses the sensory data to specify whether it is safe to continue The perception zone of this rule differs from those of the alignment and cohesion rules. The range used for the collision avoidance is less than that used in both aligning and cohesion rules, the advantage of this being to avoid overcrowding only within a local area. In addition, an agents field of view for this rule is 36 degrees; starting from 18 degrees to the right and extending 18 degrees to the left of the agents velocity vector.

For an agent A_i, located at position P_{A_i}, the centroid $C_{A_i}^{R_\gamma}$ of this rule is computed

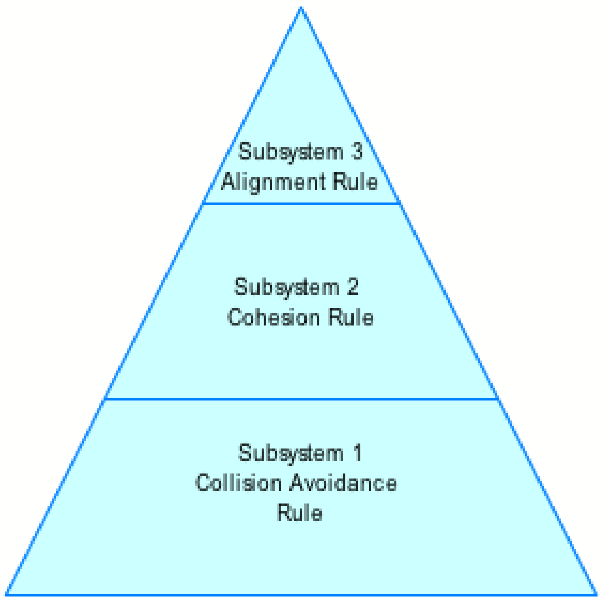


Figure 3.3: The flocking rules

as the vector to the detected object. Similar to the previously described rules, a correction angle $\theta_{A_i}^{R_\gamma}$ is set depending on the distance to the detected agent. Also, a weighting value $w_{A_i}^{R_\gamma}$ is used to give the priority to this rule over the other rules.

In the case where the agents sensor detects three objects in the three locations then it neither changes the heading nor moves until the next frame comes with a new sensory data that may show a new set of detected objects and locations. If new information continues to show detected objects in the three locations this implies the agent may oscillate, see section 4.3.4.

The flocking rules, figure 3.3, produces three different suggestions an agent needs to consider when deciding the next action. Each suggestion is comprised of a centroid, a correction angle, and a weight. The information corresponding to each suggestion is stored in the internal blackboard. The strength of each suggestion is determined by the weights associated with each rule. These weights are computed and dynamically updated each time an agent updates its sensory data. The relations between the weights and the centroids are empirically defined in chapter 5, and tables 5.3 and 5.4 show how these weighting values are computed.

To summarize, the flocking algorithm allows an agent within a team (using the sensory data) to decide on one of two general actions when an object is detected: move away from it, if it is too close, or move towards it if the detected agent is from the same team. The flocking algorithm is helpful when trying to control the motion of hundreds of moving objects which can be perceived as an organisation not a collection of individually moving agents. This is because it mimics the nature of a large organisation, examples are birds or herds of animals. Therefore, the flocking algorithm allows for self maintaining of positions and movements. However, in real life where a higher level agent tries to assign a task for this large group a concern is how to keep the user in the loop. This implies the need to integrate communication with a higher layer to receive commands for the purpose of assigning tasks. For this a blackboard negotiation technique is considered and is described in the following section.

3.3.2 Global Communication Technique: Blackboard Negotiation

Blackboard negotiation is used to coordinate the global interaction with a ground based controller GBC (the GBC could be an user). Agents in turn can regularly write messages to the GBC regarding their current status and findings. The messages are exchanged through the external blackboard. Message exchange through the blackboard is known as blackboard negotiation.

Recall that the blackboard comprises two parts; an internal and external blackboard, the internal-blackboard, described in section 3.2.1, organizes the inputs from the sensory data and the agents subsystems' outputs. This part of the blackboard is exploited by the local communication, section 3.3.1. The external-blackboard, in turn, organizes the global communication between agents and a ground based controller GBC, a higher level agent. Agents exchange messages that hold higher level information (of type intentions) via the external-blackboard. All agents including the GBC-agent are able to read/write messages from/to this global blackboard. This is a significant part of the communication with the world, as the agent gains partial-global knowledge by communicating and negotiating with a GBC.

Global interaction-communication allows an agent to cooperate with other agents in a way that avoids conflicts. This is achieved by grouping co-operation and conflict resolution. The grouping co-operation method implies grouping agents into teams and assigning team tasks. Conflict resolution can be achieved by allowing an agent to arbitrate between the flocking system and the blackboard negotiation using a weighting strategy. This blackboard negotiation technique is implemented as a fourth rule in addition to the three flocking rules. Since the flocking rules are used to coordinate motion, the fourth rule is used to coordinate tasks and governs the global interaction with the GBC for task distribution.

Practically, this is done as follows: at the beginning of performing any specified task:

the GBC issues a team task in the form of a target location for a set of agents. Each agent A_i in the team computes the expected distance $Dist_{A_i}^{tar}$ to a target (tar) positioned at (P_{tar}) according to,

$$Dist_{A_i}^{tar} = \|Pos_{A_i}^{start} - P_{tar}\| \tag{3.5}$$

where $\|Pos_{A_i}^{start} - P_{tar}\|$ is the Euclidean distance to the target. Next, an agent computes the number N of time intervals of length τ the agent needs to reach the target position depending on its speed Δx according to,

$$N = \frac{Dist_{A_i}^{tar}}{\Delta x} \tag{3.6}$$

Each agent has a counter C that counts the number of time intervals elapsed since the agent started the task and compares it to the expected N. Also, for each time interval τ , an agent estimates the remaining distance, with respect to its current position $Pos_{A_i}^{current}$, to the target which represents the centroid $C_{A_i}^{R_{BB}}$ of the blackboard rule.

$$C_{A_i}^{R_{BB}} = P_{tar} - Pos_{A_i}^{current} \tag{3.7}$$

Similar to the local interaction rules, a weighting value $w_{A_i}^{R_{BB}}$ is used to control the strength of this fourth rule. During all these stages the agent can regularly report its status via the external-blackboard to the GBC.

3.4 Communication Algorithm

This section aims at putting the communication rules together and specifying agent's acts. In this work, two types of agent's acts are identified. These are the physical acts and the communicative acts. An agent's physical act, seen as a physical displacement, is modelled as the addition of vectors in a physical space. Accordingly, a communication algorithm, table 3.6, expresses the communication rules that emerge from the previously described co-ordination techniques. It describes the agent's actions as movements by continuously setting a new position and heading on the basis of the new beliefs and desires that emerge from the rules, where:

- Each agent updates its knowledge about the environment at every time interval τ.
- For each τ, the set of guidance input information to an agent's control system includes: a) an agent's current velocity (heading ψ_{A_i}, speed S) in addition to the identities, positions, and headings of the detected agents, and b) the messages from the GBC that specifies the target position.
- The interpreter presents the information and produces a set of desired actions on the basis of its beliefs. It uses the information to calculate the velocity vectors (Correction angle $\theta_{A_i}^{R_k}$, Centroid $C_{A_i}^{R_k}$) and the weight ($w_{A_i}^{R_k}$) associated with each rule. These weights are dynamically computed according to the agent's belief about the priority of each rule. The set of agent's beliefs and inputs are stored in the internal blackboard.
- An agent then decides on the next action in the form of a physical act (i.e. as a movement computed as a weighted vector sum of the velocity vectors produced by the subsystems) or a message to report status to the GBC.
- An agent then passes its decision either to the actuator to perform it as a physical action or to the GBC as a communicative act via the external blackboard, see section

3.5.

- The agent updates information caught by the perception system (the sensory data, and the read messages from the external-blackboard).

3.5 Communication Protocols

In section 3.4, we saw how an agent executes the results of the different suggestions produced by the subsystems as a physical act passed on to the actuator to perform a physical movement. This section identifies the communicative acts an agent sends to the *GBC* via the external blackboard and how to formalise the exchanged messages. This section emphasise how an agent manages the exchanged messages with the *GBC* via the external-blackboard using speech acts. The speech act used in this work focuses on the content and not on the form, described in section 3.5.1. In this respect, this formalism of the exchanged messages implies embodying the agent's knowledge into its communicative acts; i.e. using the speech act in an informative sense. This implies that the basic operators and notations are simply expressing the agent's beliefs and intention. These operators are described in section 3.5.1. In addition, a specific example of an agent's conversation is presented in subsection 3.5.2

3.5.1 Encoding an Agent's Knowledge into Communicative Acts

An agent's knowledge is changeable in accordance to the environmental changes and to the perceived changes in the local agents' beliefs and intentions; known as cognitive dynamics. This cognitive dynamics leads to a dynamic configuration of an agent's mental state. Having classified the cognitive units into factual and provisional units, described in section 3.2, these units interact to form the overall mental state of an agent. The agent's mental state that constitutes its knowledge is represented as a communicative act. The semantics of communicative acts are initially defined in terms of beliefs and intentions from the perspective of each individual agent. In other words, the basic cognitive units form the following basic notation:

- The notation $believe(A_i, \Pi)$ expresses that agent A_i's belief that situation Π holds. Beliefs encode the factual part of an agent's knowledge at a given time.
- The notation $intention(A_i, \Pi)$ expresses that agent A_i has the goal that situation Π should be true.

A simple message is built up of one of the above described notations. For a more complicated message, an action expression is needed to form the message. For this, Cohen and Levessque formalism [89] is adopted to formalize the exchanged messages which is based on a first order predicate system supplemented by a number of operators. Recall that belief and intention are defined as the basic cognitive units, they form the basic operators. A set of simple operators was used within the proposed work and is shown in table 3.3.

In addition, a set of propositional attitudes such as preconditions of actions or sequences of communication actions are used to form the action expressions. According to Cohen and Levesque in [25], these action expressions are formed using the usual programming techniques: e.g. IF/THEN, WHILE loops, FOR loops. Communicative acts are then stated in terms of these action expressions similar to those introduced by Smith *et al* [90], and Kumar *et al* [56] using dynamic logic operators which supports action composition. Compared with Cohen and Levesques work, in our work, a communicative act is not treated as a complex action expression, but is treated as a view of the current mental state of the performer; i.e. a simple action expression represented by performative operators.

Table 3.3: SABN- operators and predicates

Category	Performative	Description
Factual operators (condition or predicative Position)	$believe(A_i, \Pi)$	Agent A_i believes Π
	$goal(A_i, \Pi)$	A_i has the goal that Π should be true
	$commit(A_i, \Pi)$	A_i has committed itself to carry out the action Π
	$intend(A_i, \Pi)$	A_i has the intention to carry out the action Π
	$perceive(A_i, E)$	A_i perceives object or situation E
Provisional operators (goals-Intentions that leads to the future actions	$Believe(A_i, \Pi)$	A_i believes that henceforth Π is true
	$exec(\Pi)$	indicates that action Π is executed
	$Goal(A_i, \Pi)$	A_i's goal is that Π should be achieved
	$askDo(A_i, A_j, \Pi)$	A_i asks agent A_j to carry out Π
	$wantDo(A_i, \Pi)$	A_i will be willing to carry out Π in the future
	$inform(A_i, A_j, \Pi)$	A_i informs agent A_j Π
	$request(A_i, A_j, D)$	A_i requests the information D from agent A_j
	$choice(A_i, A_i, \Pi)$	A_i has committed to oneself to carry out Π
	$capable(A_i, \Pi)$	A_i is capable to carry out Π

3.5.2 An Example of an Agent's Conversation: Speech Acts in a Sheet Form

The agents interactions within a multi-agent system does not arise in isolation, but as a part of extended communication activities. These activities can be thought of as dialogues including arguments and negotiations amongst agents. Conversation then is composed of one or more of these activities. For formalisation, speech acts comprise of a conversation specified in a sheet form described by Ferber [37], that is made up of two parts: *Format of message and Conditions for realizing an act.*

Table 3.4: (GBC) is delegating a task to a set of agents A_i where $i = 0, ..., n$

Format:
$Msg :: \ GBC : \{A_i \vert id = 0, ..., N\ \} << \quad askDo(\Pi)$
Pre-Conditions:
$goal(A, \Pi)\ \wedge\ believe(A_i, exec(A_i, \Pi)\ \Rightarrow\ \Pi)\ \wedge$
$believe(A_i, wantDo(A, \Pi)\ \wedge\ believe(A_i, \neg eventually(\Pi))$
Post-conditions, success:
$Msg :: A_i : \quad GBC << \ Request(Inf)\ \Rightarrow\ commit(A_i, GBC, \Pi)$
Post-conditions, satisfaction:
$Msg :: \ A_i : \ GBC << \ believe(A_i, Done(\Pi)) \Rightarrow \ believe(A, \Pi)$
$Msg :: \ A_i : \ GBC << \ notify(A_i, Done(\Pi))\ \Rightarrow\ believe(GBC, Done(\Pi))$
Failure:
$A_i : \ GBC \ << \ RefuseDo(P)\ \Rightarrow$
$believe(A_i, \neg capable(A_i, \Pi)) \vee$
$\neg wantDo(A_i, \Pi)) \vee$
$\models \ believe(A_i, \neg goal(A_i, \Pi))$

- Format: this part describes the syntax of exchanged messages, where a typical exchanged message can be denoted as follows:

  $Msg :: sender : listener <<< utterance >$

  The notation shown above states that each message is composed of: a speaker agent (the agent sending the message), a listener (the agent to which the message is addressed) and the utterance. Utterances are specified from a functionalist point of view and expressed in the form: performative(content). The word performative is used to designate the acts associated with the message, and the content is in the propositional form. For a more complex messages, an utterance is composed of action expressions and operators.

$< utterance >=< Action\ expression, attitude1, attitude2, ... >$

A message can perform more than one function at a particular time, for example:

$Msg1 :: Ai : Aj <<< Believe(A_i, A_k, < x1, y1 >) >$

which expresses the belief that (A_j) places (A_k) at ($< x1, y1 >$), also it assert to A_j the fact that (A_k) is actually in this position. In this example, the function could be expressive or referential or both. Because we considered the donative function of a speech act that is centered on the context of the language which guarantees sending the data related to the facts of the world. A message which performs this function therefore relates to a state of the world as a belief an agent has.

It is necessary for the addressee(s) to have an effective understanding of its communicational intentions. Therefore, when designing a communicational system it is essential to define rules as to how these messages pass on information about the world. This implies considering both: the quality and quantity, of the passed information. The quantity implies that a message should be as informative as possible, while quality implies that only true information should be given.

- Conditions: this part defines a set of conditions which are classified into two types: preparatory or essential conditions and post conditions. The former ones are usually related to the agent's beliefs and intentions; for realizing an act. The latter are the consequences whether success or failure of an act. Table 3.4 shows an example conversation in sheet form.

3.5.3 Modes of Communication Specified by the Proposed Protocol

According to the proposed protocol, modes of communication are classified into intentional and incidental communication, see Table 3.5. Where agents' communication with nearby agents, local communication, is considered as incidental communication. Whereas the global communication with a *GBC* is considered as an intentional communication. In the incidental communication mode, the protocol allows for the recipients to be unknown when sending the message.

Example of an incidental local communication is:

$Msg :: \ A_i : \ \{A_j \exists dist(A_i, A_j) < R\} << \ friend?; request(\phi_{A_j}, C_{A_j}); align(A_i, \phi_{A_j}, C_{A_j})$

The sender agent (A_i) tests the nearby agent (A_j) that falls within a distance less than the sensor range (R), if it is a member in the same team, then it requests the velocity information and tries to match its velocity vector with the detected agent without knowing in advance who is the recipient.

In the global incidental communication mode, a *GBC* can ask all agents who arrived at a target position ($\Pi 1$) to do the next action ($\Pi 2$) without knowing in advance which agents have reached the target.

$Msg :: \ GBC : \ A_i | believe(A_i, eventually(\Pi)) \ << exec(A_i, \Pi)$

An example of an intentional communication mode is a *GBC* allocating a task to a set of specified agents, i.e. the *GBC* intends to influence the mental state only for those agents who are specified by the message. The syntax of a message that is exchanged via the blackboard takes the form:

$Msg :: \quad GBC : \{A_i | id = 0, ..., N \quad \} << \; believe(GBC, A_i, \Pi)$

where the GBC conveys its mental state to all agents specified in the message syntax, while the message:

$Msg :: \quad GBC : \{A_i | id = 0, ..., N \quad \} << \; < \; askDo(GBC, A_i, \Pi)$

asks all specified agents to do task Π.

Unlike the current protocols implemented by agent communication languages, this guarantees informing a set of agents all at once without having any prior idea about who are the recipients or what their mental states are. For example, in the $FIPA\ ACL$ a sender can inform a set of agents by sending individual messages to each agent [38]. Kumar *et al* in [56] commented that in the major communication languages ($FIPA\ ACL$ and $KQML$) there is no way to send messages to a group of agents but to inform them individually. In addition, Kumar *et al* claimed that in these languages a prior knowledge of a certain mental state of the (known) recipient is a prerequisite to a sent message within a group communication. This implies the sender must know the recipient in the first place. In the second place, the sender must have a certain belief about the mental state of this recipient.

Table 3.5: Modes of Communication specified by the proposed protocol

Type of Message	Level Of Communication	Mode Of Communication
1: Agent-to-agent(s) (Incidental)	Local/direct	Multicast
2: GBC-to-agent(s) (Intentional-incidental)	Global/Blackboard	Broadcast
3: Agent-to-GBC (Intentional: Reporting)	Global/Blackboard	Unicast

Different modes of communications are essential when adopting this algorithm for real robots, see section 6.1.2. Considering the scenario where a set of small moving robots are accomplishing a planetary exploration task. The proposed protocol can serve well in this case where robots can be wirelessly networked to a static host, which is also sent to the same planet to allow robots to exchange messages, details are included in chapter 6, sections 6.1.

3.6 Conclusion

This chapter discussed the operational specifications of a new communicational system across two levels; micro and macro. The micro structure has been designated to provide the empirical specifications of a physical agent's knowledge and architecture. The macro structure specified the rules of interaction/communications between the agents.

The role of the interaction rules is to allow for co-ordinating actions and movements. The movement co-ordination is used in a local communication technique by implementing the flocking algorithm with key enhancements:

- In order to localise all computations, a perception zone is used so that only local agents are considered. The sensor range controls the degree of locality.
- Prioritising the actions considers all the weights; i.e. all the beliefs and intentions.
- Using distributed weights and rules. The conventional implementation of the flocking systems specifies homogeneous rules and weights in a way that all the agents use all the rules and the same weights which results in global centroids and orientation.

The flocking algorithm results in a flocking behaviour that takes the form of random movements of a team of agents as a unit.

For the purpose of assigning a task to a set of agents, blackboard negotiation is used to exchange messages between the user and agents. The blackboard is considered as a global communication technique. The blackboard negotiation results in exchanging information between agents and a higher layer GBC in order to gain partial global knowledge. In the case of conflicts, an agent can arbitrate not to carry on doing this task by switching off the global communication channel and then communicate only with local agents. Similarly if the agent arbitrates not to be a member of a given team, it can switch off the flocking system and communicate globally with the GBC. This arbitration is proposed so that an agent can overcome trap problems due to incomplete information. The empirical explanation of this is included in chapter 4, section 4.3.3.

As the proposed system aims at producing a global emergent behaviour from simple local individual behaviours, the common individual intentions result from assigning a team task by the user. The team task implemented in this work allows for a new definition of a joint intention. In the proposed work, joint intentions can be seen as a common individual persistent goal that an agent believes it is capable of achieving **regardless** of the other commitments whilst a joint intention defined in [27] implies that each agent is committed to its task **as long as the others are**. In this respect, a joint intention can be defined as follows: a set of agents jointly intending to carry out an action (Π) until it is individually known that the activity is successfully done or unachievable. According to this definition, a team can be formed when a set of agents possess a joint intention with respect to a specified goal. This has three implications. First, a joint intention acts as a binding force, similar to the cohesion force in the flocking rules, that brings the involved agents to a joint commitment rather than a mutual commitment to perform the task. Second, it emphasise task completion as individual agents will carry on doing the task regardless of any change in the others' commitments; seen as individual doubts, failure, trap.

The protocol, presented in section 3.5, identified two types of messages; incidental and intentional. An agent may pass any of these messages by multicasting, unicasting, or broadcasting. The modes of communication are particularly useful and are mentioned here for the purpose of adapting the algorithm to real robots (details are in chapter 6).

In order to visualise the emergent behaviour, the proposed MAS model is simulated. The simulation and simulation results are presented in chapter 4, and chapter 5. Chapter 4 presents the 3D simulations used for the purposes of monitoring and visualising the agents' behaviour whilst chapter 5 analyses the simulation results.

Table 3.6: The communication algorithm.

```
For each Time interval $\tau$ do:
—— Update the information caught by the perceptual system—-
$GetSensoryData(\ Id_{A_j},\ P_{A_j},\ \phi_{A_j}^{R_\beta})$
$\forall\ \{A_i|i=0,...,N\}$. do:
———— Produce a new set of desired actions————-
$GetNeibPosition(A_i, P_{A_i})$
$GetNeibHeading(A_i, \psi_{A_i})$
**Rule 1:Align**
$FindNearestTeamMember\ \rightarrow\ if\ \exists\ A_j\ \{A_j|Mindist(A_i, A_j) < SensorRange\}\ .do$:
ComputeAlignmentCentroid $(C_{A_i}^{R_\alpha})$
ComputeAlignmentcorrectionFactor ( $\theta_{A_i}^{R_\alpha}$)
**Rule 2: Cohere**
$FindLocalTeamMembers \rightarrow \forall\{A_j|dist(A_i, A_j) < SensorRange\}\ .do:$
ComputeCohesionCentroid$(C_{A_i}^{R_\beta})$
ComputeCohesionCorrectionFactor $(\theta_{A_i}^{R_\beta})$
**Rule 3: Collision Test**$\rightarrow\ \forall\ A_j\ \exists|(C^{R_\gamma} > Separation)$
if $\exists\ A_j \Rightarrow\ \ Avoid(A_i, A_j)$
**Rule 4: Read/send Messages from Blackboard**
$If\ \exists\ Request(GBC, A_i, tar)$,do:
ComputExpectedDistanceTotarget $(D_{A_i}^{tar})$
ComputeCorrectionAngleToReachTarget $(\theta_{A_i}^{R_{tar}})$
ReportStatus$(A_i, tar, GBC)$
———————— Decide the Next Action————-
**1- Produce a Physical Act: VectorSum Rules Outputs**
$X_{A_i}^{\tau} = \sum_k w_{A_i}^{R_k} C_{A_i}^{R_k} \cos\left(\theta_{A_i}^{R_k}\right)$
$\Upsilon_{A_i}^{\tau} = \sum_k w_{A_i}^{R_k} C_{A_i}^{R_k} \sin\left(\theta_{A_i}^{R_k}\right)$
**ComputeDesiredVelocity**
$\zeta_{A_i} = \arctan\left(\Upsilon_{A_i}^{\tau}, X_{A_i}^{\tau}\right)$
———————— Pass the Decision————————
**Pass Physical Action to Actuator**
SetNew(position,heading)
**Pass Communicative Act to the** $GBC$
WriteMessage()
```

Chapter 4

Visualizing Agents' Negotiations

Simulation is a very efficient tool for analysing the properties of a theoretical model. Due to the complexity of designing a multi agent system (MAS) model, simulation can reduce this complexity by building and testing the system components separately. This step by step building strategy allows the user to test and evaluate individual components during the development stages.

In order to benefit most from simulating a MAS communication model before creating a physical implementation, the simulation must give an insight into the operational requirements. Such requirements include the size of the models and the space they move in. Therefore, visualising the simulation of the MAS model within a virtual environment (VE) can actually be more efficient and functional than within conventional simulations.

Simulations within a VE supports simulating the real physical sizes and spaces. In addition, it allows the emerging characteristics to be empirically defined, tested and explained by considering both quantitative and qualitative outputs. The performance of a MAS can be evaluated and assessed within different simulated environments. This implies that agents can be exposed to a variety of different tasks; i.e agents can be created and their performance tested in different environments without an excessive amount of development time. A developed multi-agents system can therefore be used as a testbed for different tasks; e.g. in the deep sea or on a planetary surface. In this respect, the considered simulation tools must: a) give a deep feeling of distances and dimensions on the appearance of objects (geometrical realism), as well as a high level of realism of the agents' interaction as a group behaviour (behavioural realism). b) define the specifics of the application, simulating a sensor within the artificial visual system, and c) support user-interaction with the agents inside the simulated world (presence and immersion). These features can be obtained by using 3D simulation tools that can be operated on a desktop as well as within a semi-immersive projection theatre. Hence, this chapter is organised as follows:

- The 3D-representation of the scene and the objects attached to it, including the representation of an artificial visual system for each agent, and the 3D-representation of the agents' actions, are discussed in section 4.1.
- The real time 3D simulation tools used in this work for both the desktop and for semi-immersive projection, are described in section 4.2.
- The user interaction with the agents in the virtual world is discussed in section 4.2.3.
- The qualitative visual assessments and the results of evaluating the observable behaviour via a set of experiments are described in section 4.3.

4.1 A 3D Representation of the MAS model

In order to simulate the communications between agents within a MAS model to mimic a real world, it is essential that the representation of these agents and their actions have a high level of realism. The realism of this simulation is critical in ensuring that the implemented algorithm during the simulation can be feasibly transferred to a real robotic application. Simulating a MAS model, as a geometrical physical model as defined in section 3.4, implies that we have to represent the environment and the agents with geometrical shapes, which is termed geometrical realism. At the same time, it is believed realism is important to gain an increased intuitive understanding of the agents behaviours inside the VE. Therefore, great attention is paid to the behavioural representation, which is termed a behavioural realism.

4.1.1 3D Representation of the Agents and Environment: Geometrical Realism

When I first started to build the theory of MAS communication, a 2D simulation was used and implemented in Java, see appendix A. The screenshots presented in this appendix show groups of agents represented as abstract shapes, indicating the minimal information required that of location and heading. This can be useful to test the theoretical models with less attention to the practical aspects required to truly implement these models, i.e. the dimensions of the objects and the environment, including the geometrical and physical distances and spaces.

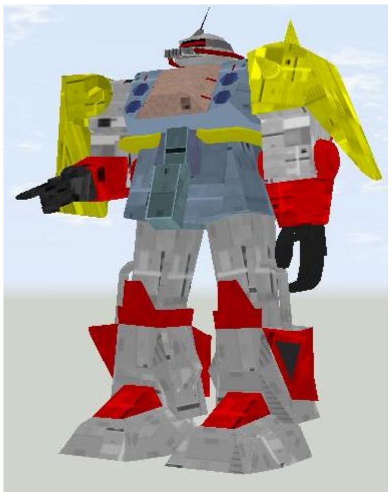


Figure 4.1: A 3D representation of an agent.

For the purpose of properly testing the *MAS* communication model, a more realistic geometrical representation of the objects and the environment was required. Geometrical realism defines the extent to which graphical output represents the real world objects. In this respect, geometrical realism is considered as a part of the design process. For example, to examine the impact of how dimensions of objects influence their behaviour, or to prototype possible sizes of the objects. Examples of 3D objects representations, shown in figures 4.1 and 4.2 , even if not accurately representing real robot design they are a good enough 3D representation. The objects represent both static (i.e. obstacles) or moving (i.e. agents) objects. Figure 4.3 shows some additional added obstacles including gaps in the simple maze.

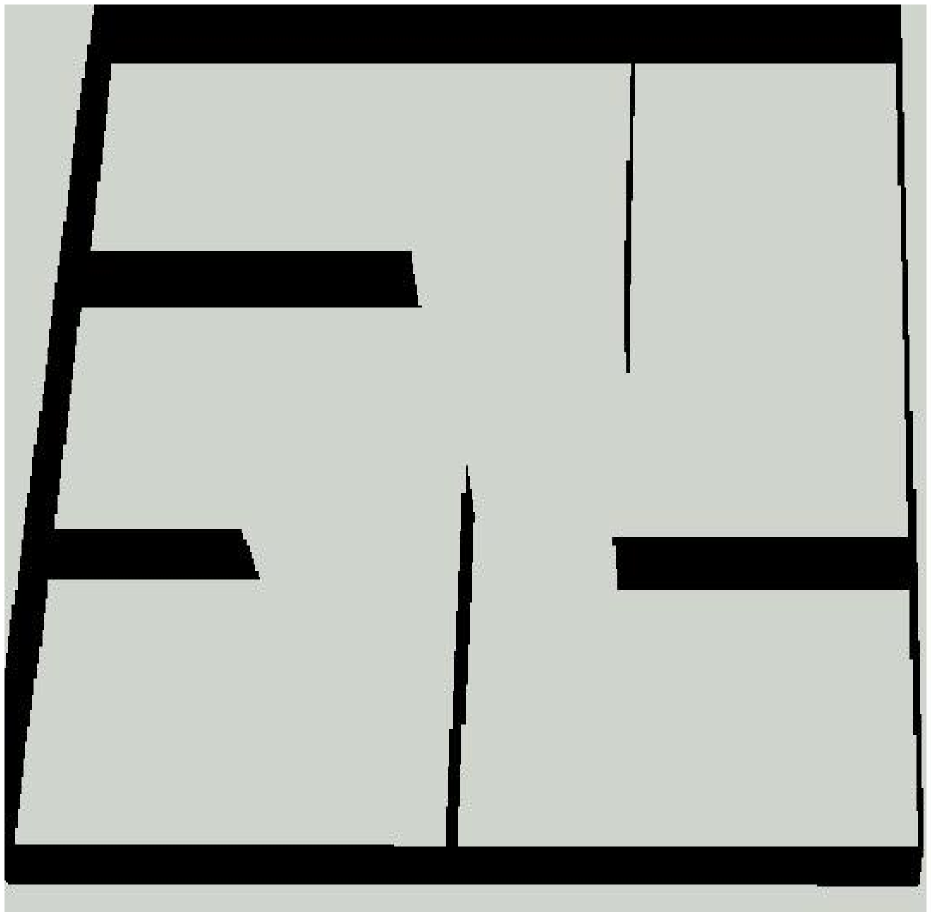


Figure 4.2: A 3D representation of the environment, a simple maze.

4.1.2 Simulating an Agent's Artificial Vision System

Another issue that is required in a real-time 3D simulation is the ability to focus on the specifics of the simulation; e.g. simulating the agent's vision system as a laser sensor. This aims at giving a direct link to the implementation of a real life robot's sensor.

The agent's sensor is simulated as a fixed line segment at a defined angle centred at the agent's location but starting just outside of the agents physical space, see figure 4.4.

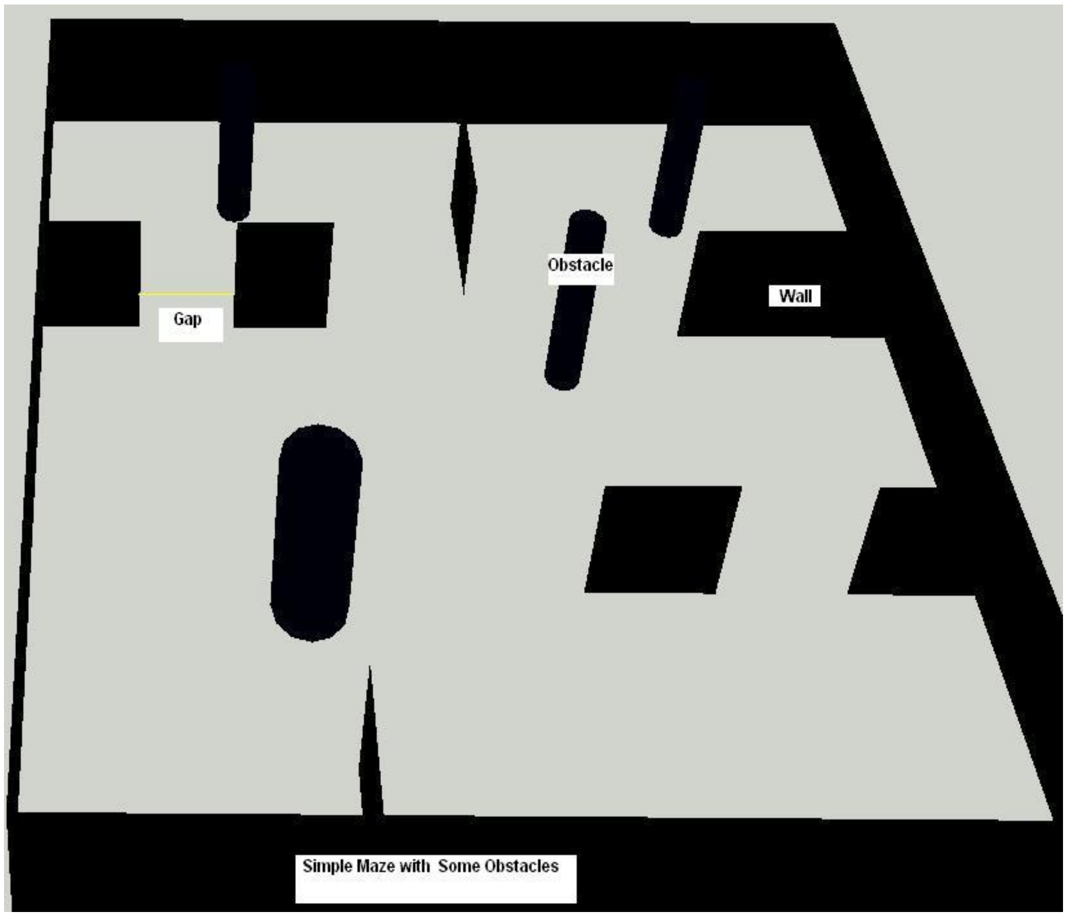


Figure 4.3: The simple maze with some extra obstacles.

The length of the line segment determines the laser sensor range. The sensor can rotate 360 degrees quantized into 18 *degree* intervals and therefore it detects one of 20 different locations at each time interval (τ).

Figure 4.5 shows that at each location, the sensor detects only one object. In this case, the sensor will only detect the first one hit (that is object A in figure 4.5).

The range of the agent's sensor is characterised by the length of this horizontal line

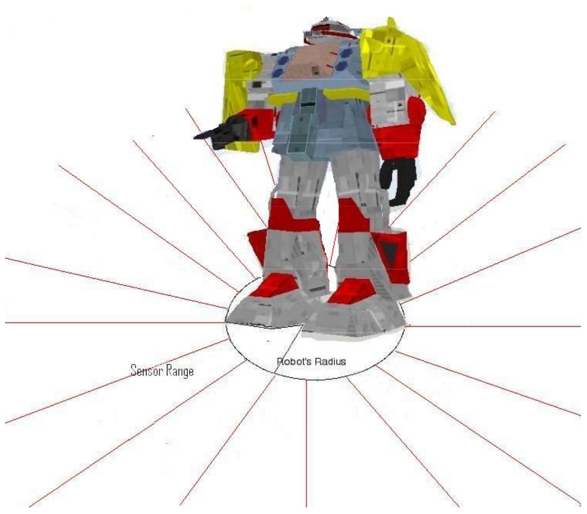


Figure 4.4: Rotating Sensor.

segment. The visualisation whether on a desktop or using the semi-immersive projection, has helped enormously in adjusting the sensor range during the very early development stages. This is aided by visually showing the sensor with colour codeing.

This horizontal sensor fired from the agent towards the environment returns the distance and the intersection point of any detected object. A full description of how the passed information by the sensor is interpreted and stored is included in section 3.2.1.

In the 3D-simulation, a user can adjust the influence of the local communications with nearby agents by controlling the characteristics of the sensor in the simulation. previous implementations of the flocking algorithms in a VE often lacked the tools to simulate the sensor that mimics real world sensors. Therefore, a goal at the very early stages of evaluation involved adjusting the range within which an agent needs to be aware of the surrounding by setting a sensible sensor range, see section 5.2.2. In order to adjust the sensor range, it is

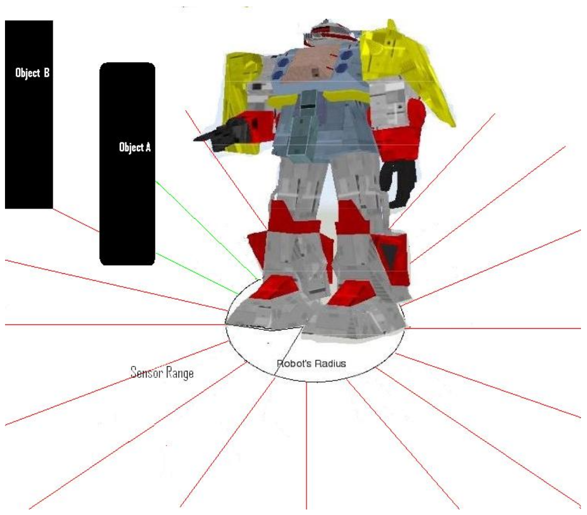


Figure 4.5: The sensor renders green once it hits an object. Object A is detected while object B is not.

important to consider the effect of detecting a relatively high number of agents during each sensor sweep and visualising the influence of the interaction rules on the agent's progress, see section 5.2.4.

4.1.3 Simulating Agents' Interactions: Behavioural Realism

Behavioural realism involves visualising believable behaviours that imitate real life reactions between agents, as a result of the interaction between their cognition units (an agent's beliefs-intentions). However, recent research hypothesised that behaviour realism within a simulation is important to aid believability [95], [23].

The creation of a believable representation of agents together with believable behaviours results in a more realistic simulation. Therefore, the main concern was to produce a behaviour model consistent with the physical appearance of the virtual robots in order to believe that the virtual robot could be seen in a manner as it would operate in the real world. The proposal aimed at implementing the three different behavioural models, section 4.3, and the main goal of this stage was to quickly evaluate each model cutting down the time required to change parameters for optimisation purposes and as an illustration to future users.

The evaluation involved running the different model on both the desktop and on the semi-immersive system. The feedback from the observers was of great importance, for example, when the $LC - Model$ was demonstrated to groups of students, they expressed their views that agents do not all communicate all the time which meets our expectation for this model. Moreover, groups of students were able to describe the actions as if they were describing real actions. In addition, when running the $LGC - Model$, the students expressed their feeling as they believed the user (me) could speak to the agents and can ask them to go somewhere very specific within the environment. They expressed an impression that the agents-user interaction was a 'real time response'.

This section has shown that realism does not only mean designing and building physically believable virtual models, but, also embodying them with believable and realistic behaviours. This helps the user to believe or disbelieve actions and provide suitable tools to interact with the agents.

4.2 Simulating a MAS Communication model within a Semi-Immersive VE

The 3D-simulation tools, introduced above, have been implemented at two levels; a desktop VE and a semi-immersive projection VE. The former implies displaying the 3D simulation at a desktop size whereas the later implies that the simulated world is displayed on a big screen. In addition to the visual simulation, all numerical values are recorded. These numerical values represent the paths followed by all agents, x and y positions, the heading angle, the new heading, as well as all the interaction weights and calculated centroids.

4.2.1 The Desktop VE

The desktop display is used for verifying the built components during most of the design stages. The software environment used for the real-time visual simulations includes: a user interface and the programming environment, (see appendix B for system details). This simulation interface provides the user with the ability to manage the set up, e.g. the appearance of the complete scene including the number of view points needed and the level of details required. There is also complete flexibility of the viewing location and the direction being monitored and the view of any action possible with multiple views. The interface enables the user to: a) control the dimensions of the environment, i.e. the sizes of the rooms and corridors, and b) define and test, interactively, the size of the target and the target model.

4.2.2 The Large Scale VE

In order to display the scene in a full scale VE, a distributed version of the desktop model can be produced through a cluster of 6-PC's to generate six synchronised images. These six images are projected onto an 8×2 *meters*, 145 *degree* curved screen (giving a $3K \times 1K$ resolution) using six different projectors. Recall that the number of views and cameras are controllable via the user interface, and all the camera views are drawn and synchronously displayed on the screen.

This large scale visualization allows for agents' movements and interaction inside the virtual world to be projected in the Virtual Environment Centre (VEC) enabling multiple users to visualize and assess the simulation at once. Figure 4.7 shows an abstract version of the VEC at De Montfort University. The space in front of the screen can host up to 20 users at once and allows for collaborative analysis to take place between experts from various fields to monitor, discuss and control a simulation. This has been used for group analysis, and for information presentations in the form of demonstration, see figure 4.6. The demonstrations aimed at displaying different models and versions have been used to obtain feedback from non-experts, i.e. undergraduate students from different disciplines.

4.2.3 User's Interaction with the Agents in a VE

In the desktop and the large scale VEs the user can:

1. Interactively choose a path to move and fly around any object in the environment.

2. Choose different starting points when issuing commands.

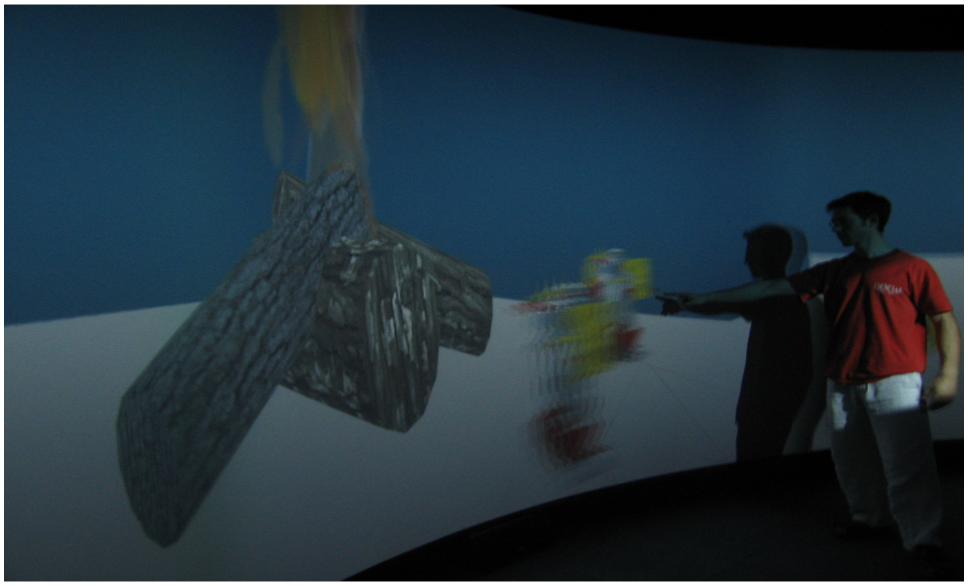


Figure 4.6: A screenshot, photograph taken while running a large scale system, demonstration to a group of students.

3. Initiate new tasks or simply define new targets and monitor, closely, the agents reactions and changing behaviours.

4. Issue a new command for a new team; initialised by, for example, grouping two existing teams together.

5. Interactively monitor the agents' responses. This improves the user's impression of being immersed in what is seen.

Unlike the animated scenes in recorded movies, these interactive aspects improve the overall feeling of being inside the simulated world. This implies providing the user with the ability to fly and walk around the objects, for example to monitor what is happening behind a wall or under a table. In addition, the user can zoom into the scene when a precise

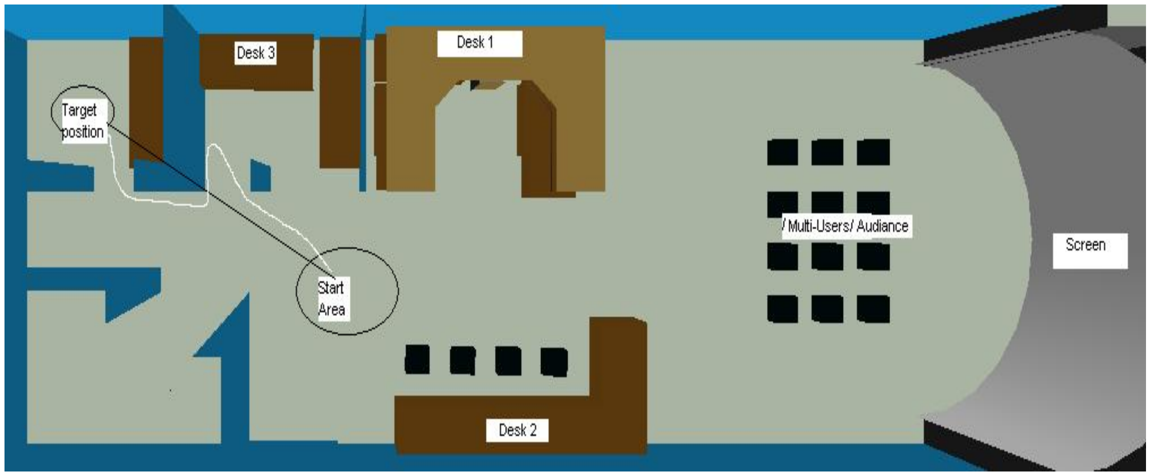


Figure 4.7: Virtual laboratory with different starting points and target positions marked.

local evaluation is needed or zoom out when a global view is required. Presence, described in section 2.3, is one of the key outputs of the VE visualization. The reason for this is that presence in the simulated environment played a major role in designing the experiments described in section 4.3. A key aspect in the experimentation process is to assess agents' behaviour by integrating all the visual outputs. Both aspects, presence and immersion, required a high level of realism in the simulation and a certain minimum level of quality for display.

4.2.4 Advantages of a Large Scale Simulation

Simulations within virtual environments are a major interest within this work because of the additional features that can be obtained using semi-immersive full-scale environment VE. The big advantage over a pure simulation is the building of an environment that

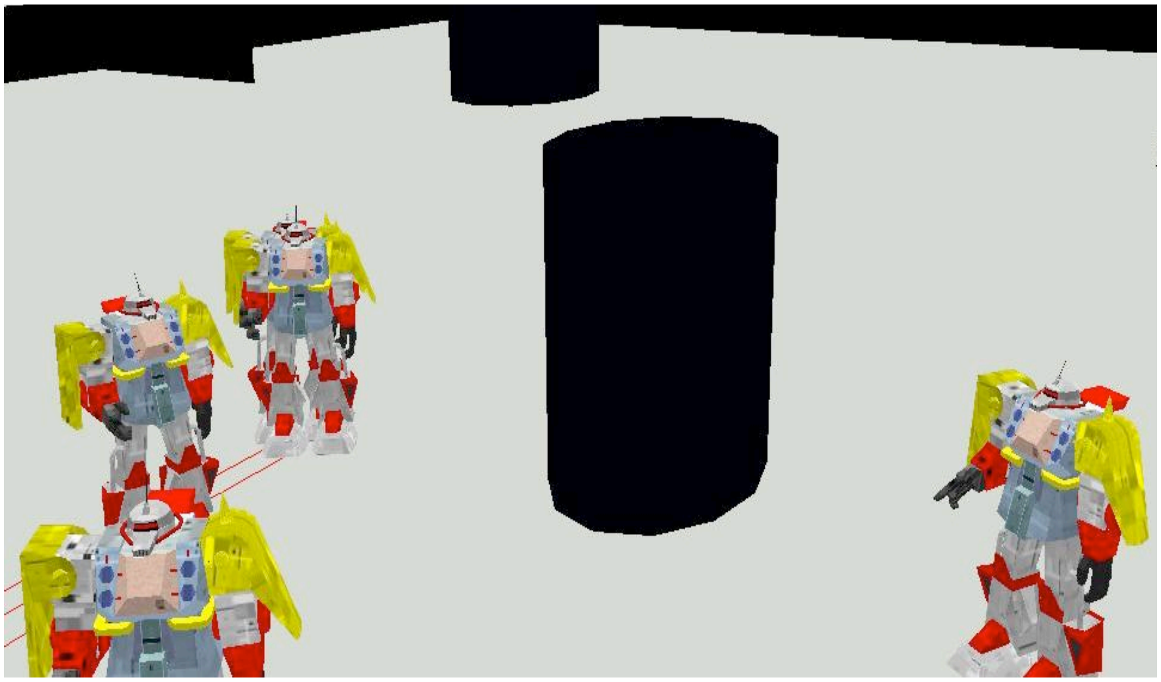


Figure 4.8: Viewing agents' interaction; zoom in.

mimics, admittedly to a different quality level, the real physical space, see figure 4.10. This full scale visualisation enables us to test different situations with different scales and dimensions and can enhances the sense of the dimension and sizes of objects inside the environment, see figure 4.11.

The large scale display also offers the opportunity to visualise multiple views at once, see figure 4.12. This allowed for a quick visual, individual and group, analysis for the quality of the agents' actions in the virtual world for this large number of agents by allowing us to move closely to the individual agents in different situations.

Another issue that is considered as an advantage for using the 3D simulation within the VE is the ability of increasing the appearance of scene complexity whilst reducing the

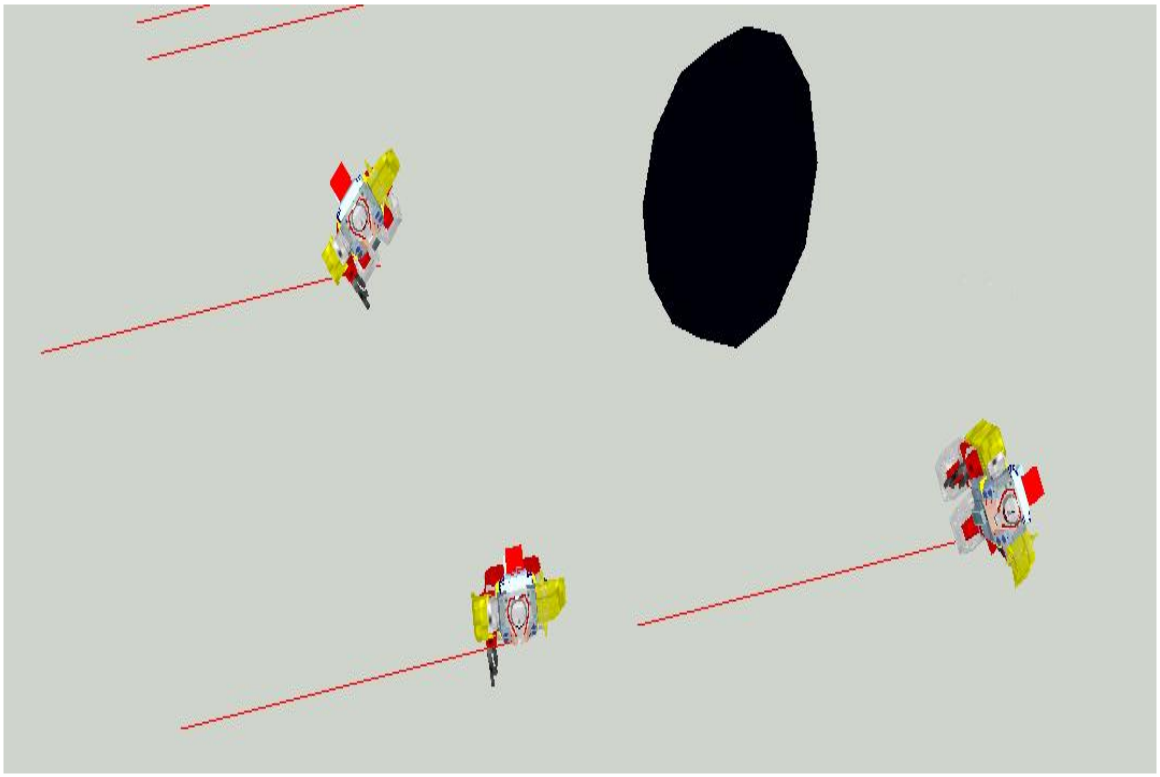


Figure 4.9: Team members split into teams; zoom out.

time-cost of always rendering the highest details. Since the object representation inside the world does not necessarily imply representing all the details all the time, the simulation in a VE allows the system to render different level of details according to the distance from the viewpoint. The higher detail geometry is rendered when objects are close to the eye point and only lower detail geometry is rendered when objects are further away. It also allows for multiple levels of detail, where there would otherwise be abrupt changes.

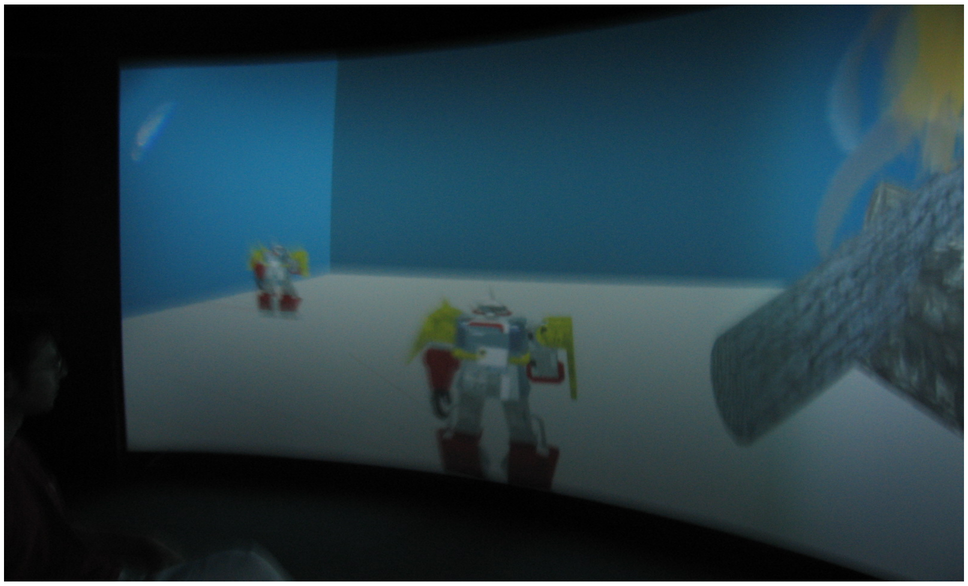


Figure 4.10: The large scale simulation supports real physical sizes and spaces.

4.3 Evaluating the Emergent Behaviours: Visual Analysis

Significant benefits of using interactive 3D-simulations only occur if the VE assists the user in achieving objectives more efficiently than other simulations would. The simulation tools used in this work allow the human vision system to assess an agent's behaviour by running the system on a desktop or projecting the scenes onto a large screen in order to improve the ability to investigate many more design alternatives using different micro structures, that implies testing different agent's architectures. Therefore, the evaluation process focuses on visually assessing the quality of the agents' actions and tests different behavioural models to see whether they meet the user's expectations. Visualising different levels of behaviour under different conditions allows us to define 'what-if' situations, to discuss times when the system fails, limitations on the number of agents, and sensor range

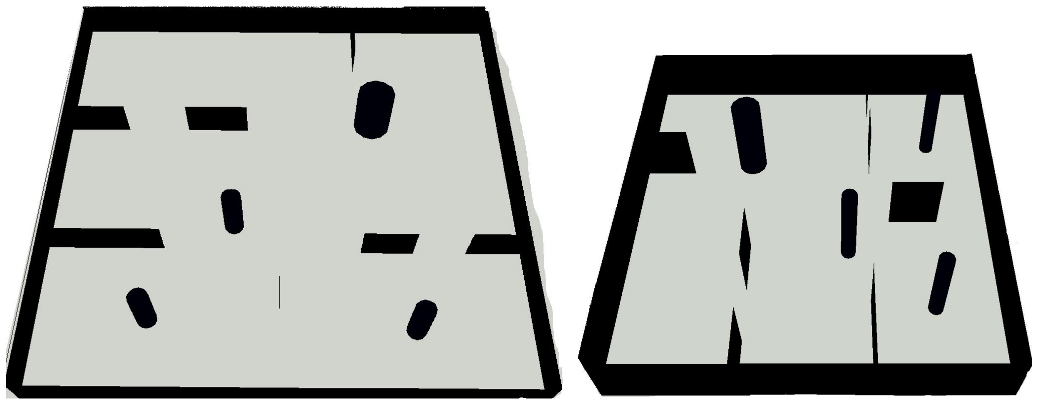


Figure 4.11: The different scales give different representation of the same spaces.

modifications, etc. It provides us with a way to explain the cases that are considered as bottlenecks. This improves the performance by evaluating the emergent behaviour at the system level not just at the individual level over a number of iterations.

In this respect, the visual assessment considers two main issues. First, to what extent the agents actions are consistent with their knowledge; where rationality is considered as a measure of the consistency. This is carried out by comparing the numerical and written messages sent by these agents with the observable actions of these agents. For example, a message of content: 'I am agent A_i at position x, y and can see agent A_j 18 *degrees* to the right within the avoidance zone'. The user expects this agent when appropriate to move 18 *degrees* to the left in order to avoid the detected agent. Second, the quality of the performed actions and the decision made are feasible of being performed in real-time; that is they are guaranteed to behave within a certain time.

The experiments combine the local and global communications into four layers. These

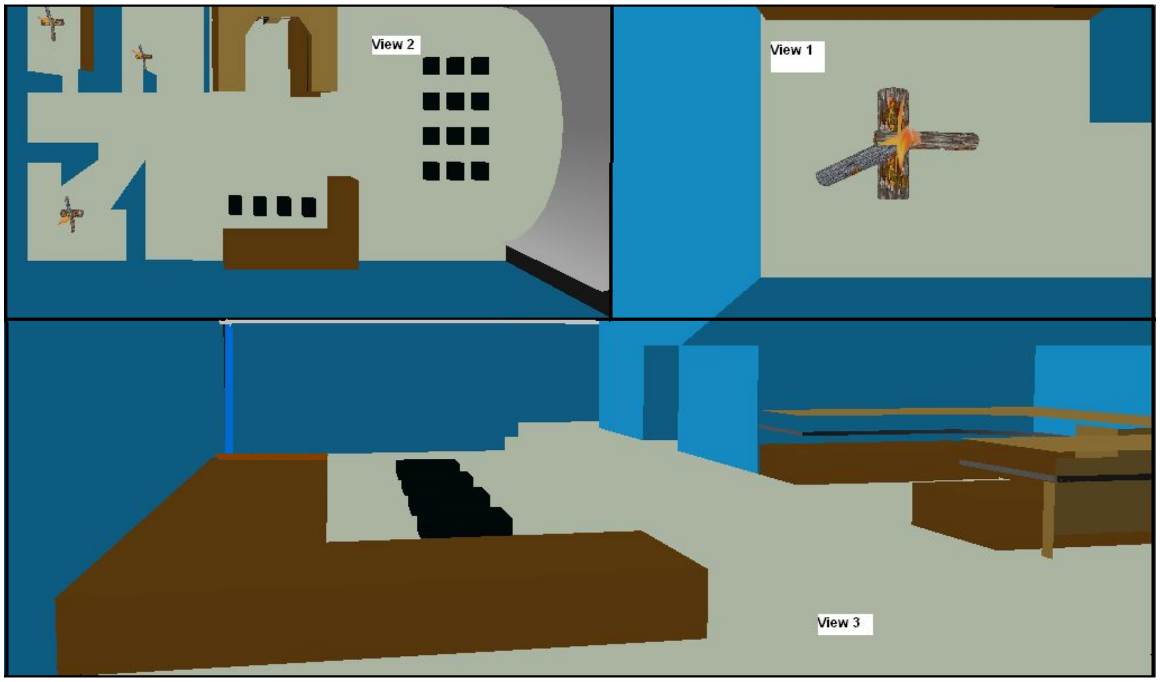


Figure 4.12: Leicester Reality Centre, De Montfort University, displaying multiple views simultaneously.

layers encode the different levels of behaviour and agent's architectures within different models: Local communication Model, ($LC-Model$), Global Communication Model ($GC-Model$), Local-Global Communication Model ($LGC-Model$), and Follow Wall Model ($FW-Model$). The following subsections describes the implementation of each of these models, and by visually testing the agents' actions, and the outcomes are compared to the user's expectation from each model.

Figure 4.13: Viewing a large number of agents, photograph taken while running the large scale system.

4.3.1 Local Communication Model ($LC - Model$)

This model implements the flocking rules without any global interaction or knowledge. The $LC - Model$ represents an initial case where agents are expected to only communicate with nearby agents to coordinate movements. Therefore the only inputs for the agents in this model is the received information via the agent's sensor and they are not assigned any tasks. The expected behaviour from this model is to obtain the natural behaviour of a flock.

The agents are expected to show this reactive behaviour that represents animal intelligence in avoiding each other and moving in a group.

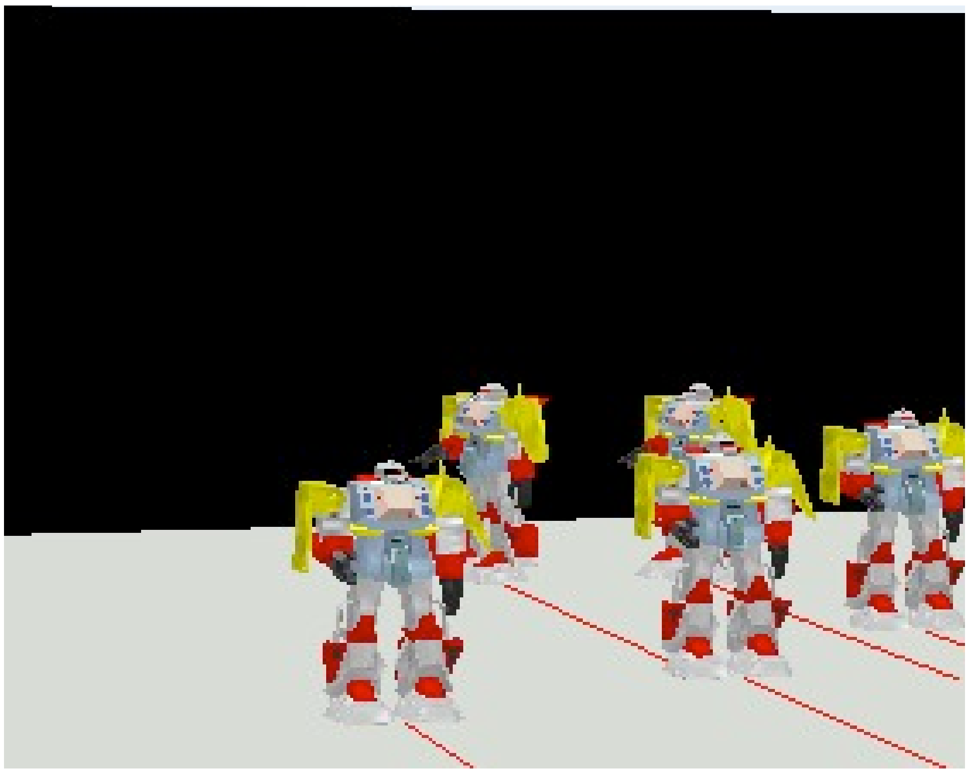


Figure 4.14: A set of agents are moving within a team and influenced by the cohesion force and the alignment force.

Running this model results in a lower level of behaviour and allows for the reactive components of the agents to be active. This model allows us to test the system's micro structure; i.e. the reactive component of the agent's architecture.

The tests show a high level of co-ordination as agents were able to move in groups with signs of natural behaviours. For example, the flock can split up into two smaller flocks to go around both sides of an obstacle and then rejoin once past the obstacle. In addition, a

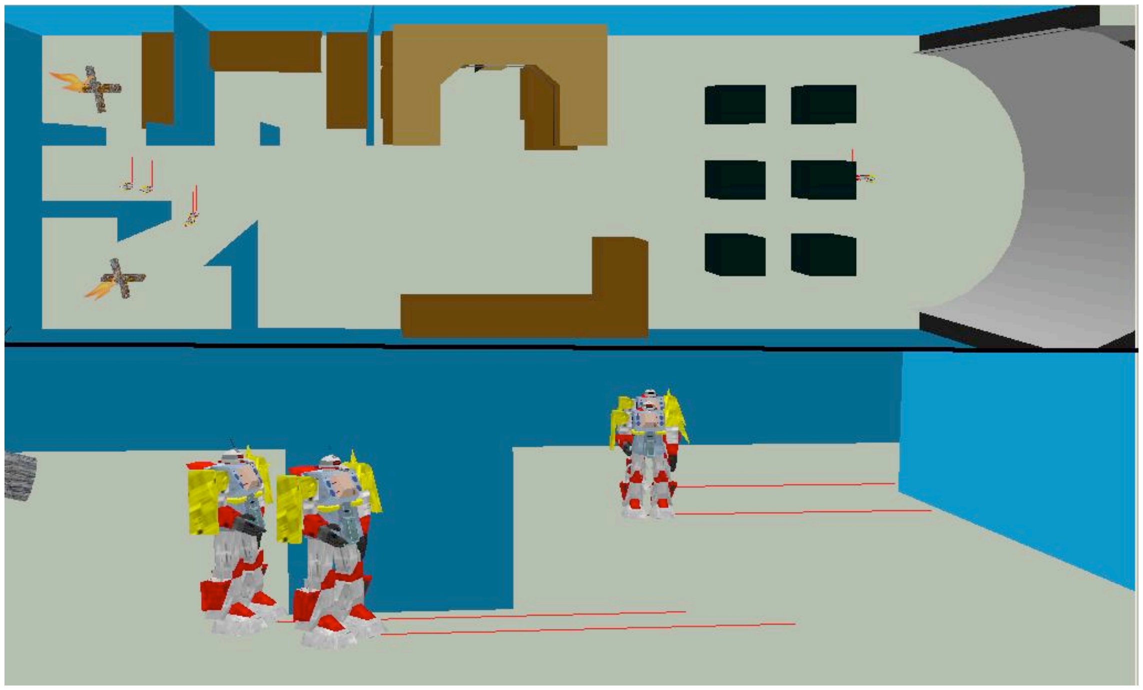


Figure 4.15: A team of agents moving forward, the team members split into two teams as they encountered the wall.

dynamic leadership is also achieved as any front agent can lead. So, if the leader should get trapped between other robots, any front agent will take over the leadership and the nearest agent will follow. Figure 4.14 shows a set of agents represented as 3D-robots. They move randomly following the first three rules of the algorithm. An advantage of implementing the flocking algorithm is that agents move through different routes, see the xy-plot in figure 5.3. This is quite useful for some applications such as passive searching or rescuing tasks.

4.3.2 Global Communication Model ($GC-Model$)

For this model the ground based controller (human) broadcasts messages, via the keyboard, to all registered agents within the system. These messages include explicit instruc-

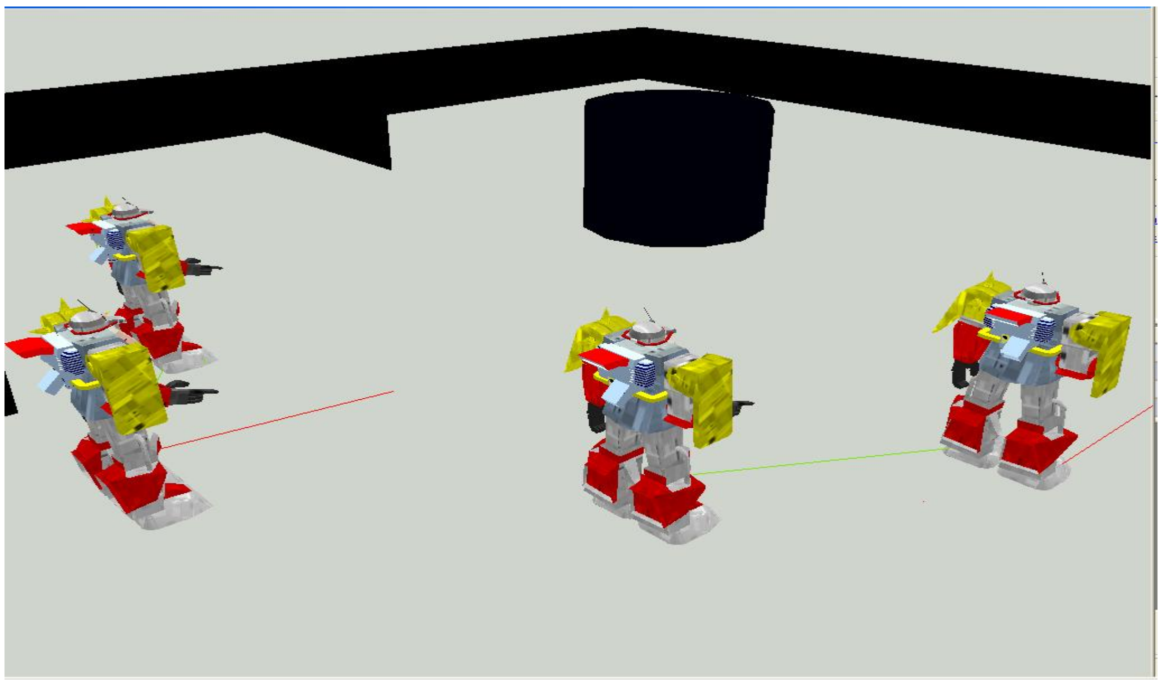


Figure 4.16: Viewing agents' interaction in a large scaled model.

tions to allocate tasks for a set of agents. When these agents read the message, they are grouped into a team; the concept of team forming. In terms of the flocking rules, each agent only considers the collision avoidance rule. In this context, an agent's internal system sets the weights that control the strength of both alignment and cohesion force to zero. The effect of applying global communication in this way is equivalent to the effect of applying a global alignment force, i.e. all agents will align their velocity in the direction of the target position.

According to this model, an agent is expected to activate only the cognitive component of its architecture. This component allows an agent to possess a cognitive knowledge structure and in this case each agent of the group is supposed to possess a joint mental state that

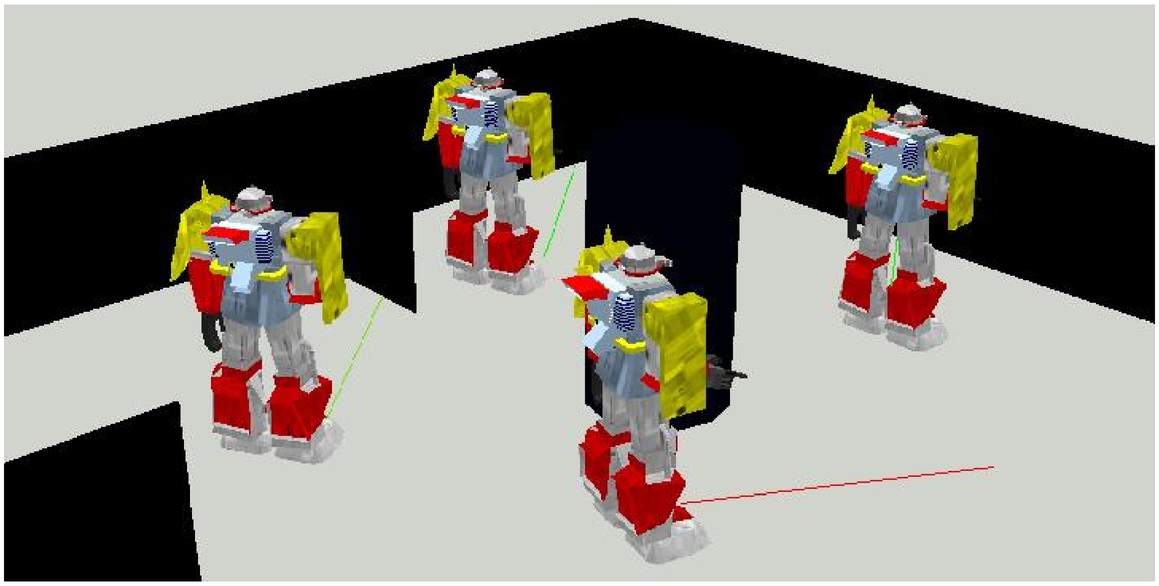


Figure 4.17: Viewing agents' interaction in small scaled model.

leads to a joint intention; i.e. performing the common goal.

This model is tested by running the system with a specified number of agents, (five agents for the first trial), and allowing the *GBC* to interactively issue a message to the agents using the keyboard as an input device. The agents inside the virtual world receive the message and interpret the contents of the message. The agents first issued messages to the *GBC* to report their status at each time interval τ. For example, an agent is happy to perform the task if $(T <= C)$ [1], also it is a little bit "Tired" if $(T - C <= 20)$ [2]. If $(T - C \approx 1200)$, this means the elapsed time of about (2 *minutes*) since difficulties started to arise, but still an agent will try to perform the task. But if $(T - C >= 2000)$ then it is likely to be trapped somewhere, and status 'lost' is reported to the *GBC*. If the

[1] C is the expected time for task completion, while T represents the elapsed time since the agent started the task.

[2] $T - C$ is the time elapsed since the agent started to encounter problems in performing a specified task

$(T - C >= 6000)$ implies that the elapsed time say (5 *minutes*), an agent will report a virtual death status 'dead' to the GBC. For both 'Lost' and 'Dead' status, an agent will do the following actions: It reports its status to the GBC via the external-blackboard and discards the specified task; sets the weight of this rule to zero. It then moves randomly until it receives an alternative command from the GBC, who can monitored the situation. This implies that an agent is unaware of other recipients of the same message. The test has shown that an advantage of this is that the action may not be discarded from all the team members.

4.3.3 Local-Global Communication Model ($LGC - Model$)

According to this model, agents are expected to be able to communicate with each other as well as with the ground based controller GBC; i.e. all the communication levels are implemented. All agents are supposed to be grouped in teams as a consequence of issuing a group command by the GBC. So this model exploits agent's reactive and cognitive components represented by the HBDI integrated with the BB in one architecture.

The test considered the scenario where a set of agents are assigned a task, e.g. they are given a target location, in the upper left in figure 4.7. By assigning the virtual task, each of the agents in the group is expected to: a) compute the expected distance to the task, b) estimate the time to finish the task, and, c) issue messages to the GBC reporting these information as a status 'HAPPY'. Each agent is expected to report its status depending upon its dynamically changing beliefs. The reported status is a result of activating the deliberative component of the agent's architecture together with the reactive component.

In other words, an agent uses all the information available, not only those related to the local reactions but also those related to the *GBC*'s commands. On arrival, agents are to circle around the target and continuously modify their weight for the blackboard negotiation rule, in order to avoid getting too close to the target.

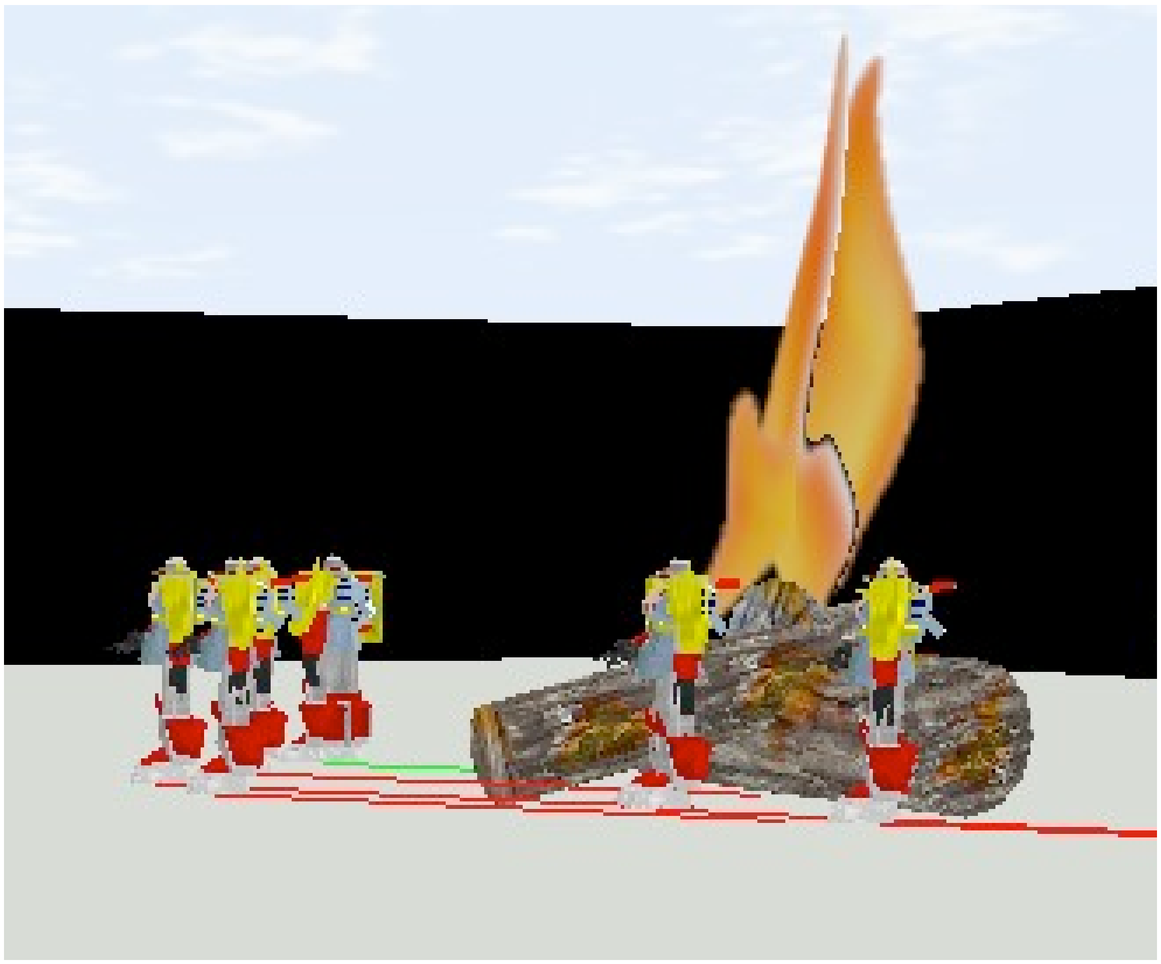


Figure 4.18: Team members are circling around a target position.

Running this model has shown interesting results across the micro and macro levels. Regarding the micro structure, the agent's actions have met the expectations where the deliberative component in the agent's architecture works synchronously with the reactive component. This test allows us to compare the different behaviours based on different knowledge structures. The emergent behaviour depends on the cognition intelligence as a prerequisite for the deliberation, see $GC-Model$. Regarding the macro level structure, the visual tests assists in building the new model that reflects the features of both models into one, namely the $LGC-Model$.

The visual tests played a major role in evaluating the group communication as it allows for interactively monitoring the agents' actions in real time. The four layers of communication (three layers for the flocking algorithm plus one layer for the blackboard rule) are all implemented.

In addition to the visual outputs, the numerical outputs can also be analysed, for example, the interaction weights and positions. This information together with the visual experiments allowed us to identify the effect of the cohesion force as a binding force that affects the agents' progress, see section 5.2.4 where the user is able to numerically analyse the weights. This has been significantly useful when the visual test showed slow responses in the agent's progress when combining the flocking rules with the blackboard.

The results of running this model show that the user is able to get comprehensive information from agents in the form of actions and written messages. The user can determine the conditions that increase the chance of completing the task despite difficulties that may arise, for example, when any agent fails to perform the specified task and other team members will then continue to perform the task. Practically, this has been implemented in the same manner as described in section 4.3.2.

4.3.4 Oscillate State Detection and Follow Wall Mode: $FW - Model$

The $FW - Model$ resolves the conflict that can arise when an agent needs to reach a specified point that lies behind a barrier or a wall. The 'Wall' problem implies that an agent would turn to avoid the wall then turn back heading toward the target so leading to oscillation. This can also happen, for example, when an agent's sensor reads three consecutive similar values for the identity of the detected object within the perception zone of the collision avoidance rule, see section 3.2.2.

The basic idea behind the $FW - Model$ is to let a meta-level component run the world model faster than real time to make predictions of future states. When the meta-level component detects that the system has specified an undesired action, it modifies the decision made to produce a new actual action in order to avoid reaching this state.

This model enables the agent to detect the oscillation state and then allows the agent to switch between two modes. The first is the standard $LGC - Model$, whilst the second is the Follow Wall Mode or $FW - Mode$. The $FW - Mode$ is used as a recovery mode and allows the agent to move smoothly alongside the wall until it can turn again toward the target. Once an agent has passed the oscillation state, it switches back to the standard $LGC - Model$.

The $FW - Mode$ is implemented as follows: Firstly, each agent performs a pre-collision detection test using its current heading. Secondly, it computes a new heading according to the communication algorithm explained in section 3.4. The agent then will use the new heading to perform a post-collision detection test before changing its current heading. An agent will not change its heading unless there is an object straight ahead. For the cases when an agent detects three different objects this model may still fail as an agent may still show oscillation whilst trying to avoid these objects.

4.4 Conclusion

This chapter first presented the 3D VE as an interactive simulation in order to observe the emergent behaviours when implementing a MAS communication model. It introduces the use of the 3D interactive simulation to enable designers to operationalize the theoretical concepts for empirical studies. It also allows the designer to test the communication model with different numbers of agents, teams in different sizes and also in different environments using both a desktop as well as a large scale virtual environment.

The 3D simulation tools used in this work offer sufficient techniques that allow for perspective views of the agents and the environment in real time. It also allows for user interaction, real time monitoring, and qualitatively assessment of the agents' behaviour. Therefore, the focus was on assessing and analysing the visual quality of the agents' actions via a set of experiments that aimed at testing the representation of the agents' behaviours as a combination of the local and global communications in four layers which encode different levels of behaviour. It is proposed that these four layers can be used to encode more complex behaviours allowing us to test different architectures and the corresponding behavioural levels.

The advantages of the large scale visualisation of the simulation were also presented in this chapter. The 3D-simulation in a VE features four main aspects: a) an increased level of presence and immersion within the simulation that helped gain an increased level of understanding of the agents behaviour inside the world; b) enabling a user interaction by flying around the agents and observing and monitoring the agents' actions in the corners, under the tables, and behind the walls, and c) supporting the realism and reality of the simulation which is of great importance for testing how this system would work in real life and what are the recommendations for real world implementations by running different virtual scenarios. The ability to simulate the real physical spaces and sizes allowed for easy fault detecting and for real-world problems to be tested and seen in an easy and understandable manner.

This chapter tested the quality of the agent's actions using the human visual system to assess the success and failure of the agents' interactions and communications. Chapter 5 presents the numerical evaluation of the simulation by a set of experiments that describes the process which sets the sensor range, the flocking weights, and the optimum team size.

Chapter 5

Quantitative Analysis

In chapter 4, it was shown that qualitative analysis can be carried out by monitoring and visualizing the agents' behaviour within the virtual world. The simulation was visually assessed to understand the emergent behaviour, and this process is of great importance to test and evaluate as well as demonstrate the system performance at all stages.

This chapter presents the second part of evaluating system performanceusing quantitative and numerical analysis. The set of experiments included in this chapter aim at testing numerically the outcomes of the agents' different behaviours based on the PTA-agent's structure. The scalability of the system is also assessed in the light of computation time. Therefore, the experiments investigate the capabilities of a set of agents to compromise and arbitrate between the local interactions and the global communication with a user. The aim is to move a team efficiently towards a specified location. An extra benefit shown in these experiments is the observer's ability to detect the location of complexity in the environment by analysing the interaction weights and the positions of the agents.

5.1 Scalability

What is meant by scalability? Is the system limited to a small number of agents and at what level can it scale up? Often scaling up to large sizes can introduce problems of computational efficiency. Real-time computation is considered important for monitoring purposes. The scalability of the multi-agent system model, as defined in [105], is a linear relationship with computational time. The computational time is measured in terms of the time required for the CPU system to perform the decision making computations for the set of agents to complete a virtual task. This includes individual computations performed within the inner loops in addition to the internal processing time required to render the scene for each frame. Allowing the system to run as fast as possible, the computation time is calculated as the number of frames, required for a set of agents to complete a task, divided by the frame rate; which gives us an indicator of computations per second, see Table 5.1.

In order to assess the scalability of the system, the experiment compared the computation time averaged over five rounds, for different population sizes starting with five agents and scaled up to fifty agents all carrying out the same task. The experiment ran the $LGC - Model$ using the virtual lab as a simulated environment. In addition, the (GBC) issues the same task from the same starting point for each trial and round. The averages for the trials for the five rounds are presented in table 5.1 and the graph in figure 5.1, which shows a linear least square fit for the computation time as a function of the population size,

Table 5.1: Scalability.

Population Size	Frame Rate (Frame/sec)	Computation Time(sec)
5	30	50
10	19	95
15	14	150
20	11	213
25	10	261
30	10	298
35	9	356
40	9	393
45	8	450
50	8	512

with the coefficient of determination $R^2 = .9645$. Statistically, this coefficient is used as an indicator to reveal how closely the estimated values for the straight line correspond to our actual data values as the R^2 value ranges between 0 and 1. A linear fit then is most reliable when the resulted R^2 value is at or near 1, therefore, the value of $R^2 = 0.9645$ is a very good fit for the line. This implies that the system effectively scales up to 50 agents. Increasing the number of agents more than 50 *agents* leads to a non-linear relationship as shown in figure 5.2.

To conclude, the experiment's results showed that the system scales up efficiently for a certain number of agents and it can handle increasingly complex structures of groups that demand a relatively large amount of self-organization and motion coordination without any direct change to the underlying mechanisms of the algorithm.

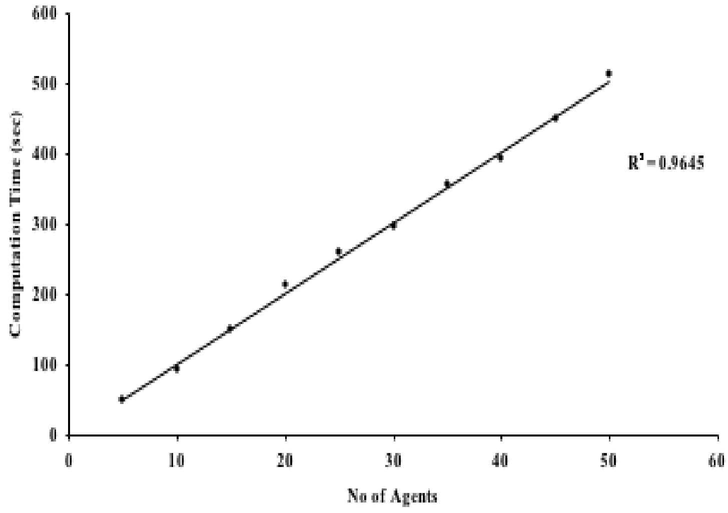


Figure 5.1: The computation time as a function of the population size.

5.2 Analysing Emergent Behaviour for the Flocking System

5.2.1 Advantages of Flocking System: Routes and Coverage Areas

An attempt is made to explore the influence of the flocking rules on the final emergent behaviour. This is carried out by viewing two values, the positions of the agents during movements and the area these agents cover after arriving at a specified target. These two issues can be especially useful when the agents are to perform a search or sweeping task, where there is an aim to visit many points whilst they are moving in a team.

Therefore, an experiment is designed that considered both agents' local and global communications. The $LGC - Model$ is used for this experiment, with a set of five agents forming one team. After launching the model, these agents are issued a team command

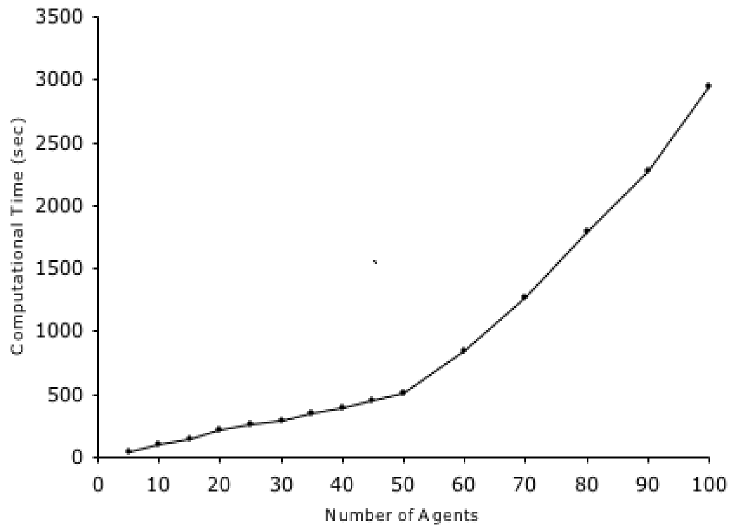


Figure 5.2: The computation time as a function of the population size up to 100 agents, note the non-linearity as the population size becomes bigger than 50 agents.

that informs them of the target location. The positions of these agents are recorded and by plotting the positions of the team members over time, it was found that agents within a team select different routes on their way to the target as illustrated in figure 5.3. Another advantage of the flocking algorithm is that, on arrival, agents within a team will cover a wider area around the target position. This prevents the agents from overcrowding the target location and they are shown to appear to circle the target. This can be seen in figure 5.4, showing all the positions of the set of 5 agents running the $LGC-Model$ over a period of time. These show how those agents swarmed about the location. The region of the covered area is computed as the number of occupied cells in the grid, each cell represents ($25\ cm \times 25\ cm$). This implies that as the number of occupied cells is 17, the agents cover 1.0625 m^2 during the last 6 frames. Comparing these results with those resulting

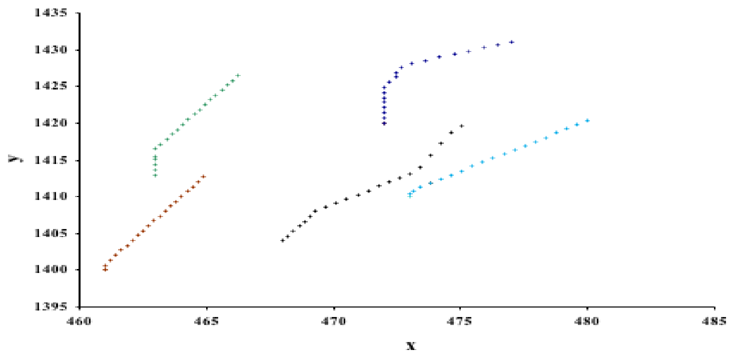


Figure 5.3: The agents select different route whilst they are moving in a team.

from running the same number of agents by the $GC - Model$ in figure 5.5, the number of occupied cells is 9 *cells* covering only 0.5625 m^2. This indicates that the coverage area by the flocking agents is about double that covered with individual agants.

The effect of the flocking system in the above results can be explained as follows. Running the $LGC - Model$ implies that any agent is able to interact locally and globally. This means, all the agents are committed to move within a team to reach the target location, and correspondingly, the team centroid is moving towards the target. On arrival, as the first agent moves forward the team centroid, which affects the cohesive force, also moves forward. Accordingly, the other agents who detect this agent also consider this team centroid in each calculation which implies they are pulling each other forward and at the same time towards the target location. These local interactions lead to the progress of the team centroid which in turn leads to the movement of the detected team members as rolling around the target location.

On the other hand, by running the $GC - Model$ the agents intend to reach the target

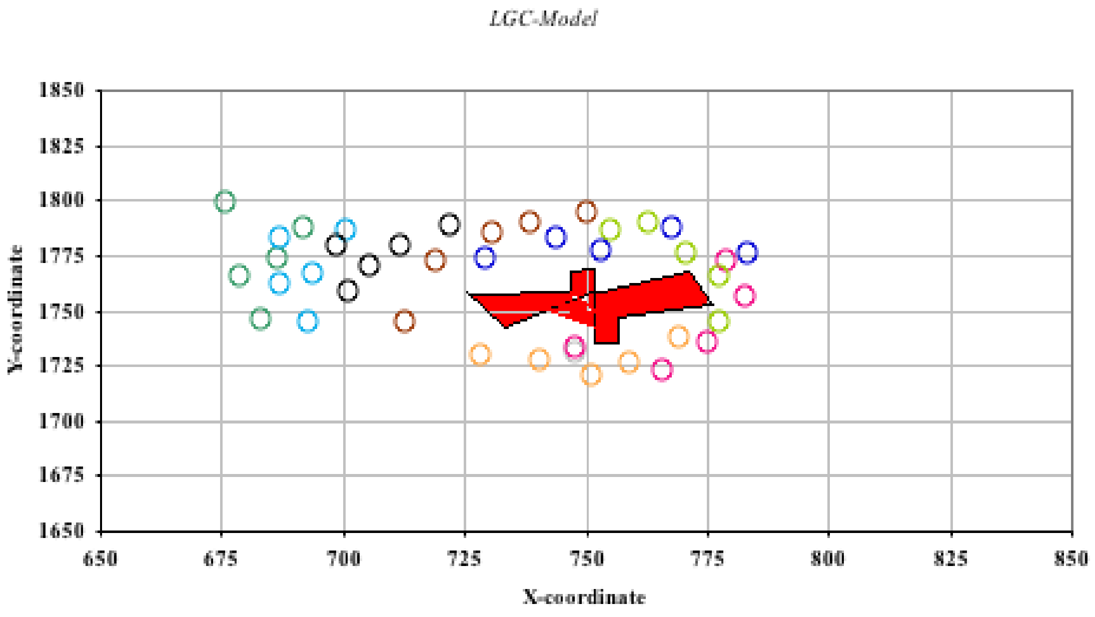


Figure 5.4: The flocking behaviour supports maximising the coverage area.

and only check for collisions. Therefore, these agents, on arrival, are either trying to avoid each other or looking towards the target which leads to a reduced possibility of covering a larger area.

The question now is to what extent this flocking behaviour is required to gain system performance. This implies considering the inputs to the flocking system and the weights that controls the influence of each rule in the flocking system. As explained in section 3.3.1, the only source of inputs to the flocking system is via the sensor. So the sensor range and direction characterise the perception field within which these rules operate and contribute to the agents emergent behaviour. In addition, the weights corresponding to these rules control the strength of these rules and are also dependent on the sensory data. At this

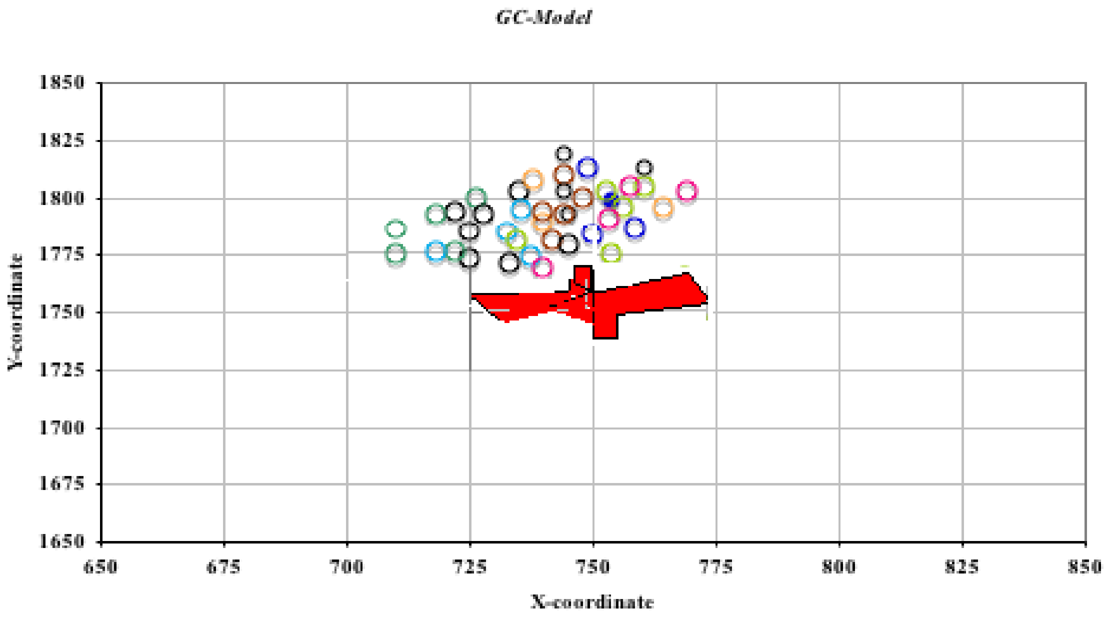


Figure 5.5: The covered area is less with the flocking system switched off.

stage the main concern was to find the optimum sensor range that enables the agent to compromise between the local interaction and the global communication demands.

5.2.2 Sensor Range SR

As the sensor links an agent to the environment and controls its local interactions via the flocking rules, this section aims at testing the optimum sensor range that allows agents to: a) move and act as a team, and b) minimise the number of frames to complete a specified task. The number of frames indicates the number of steps, and consequently the distance an agent moves in order to reach this target. Completing the task means being within a distance equal to double the sensor range from the specified target. When an agent moves within a team towards a target, this implies that it activates both the local interaction rules together with the external blackboard rule. The local interaction rules require an agent to filter the sensory data according to the minimum separation distance allowed between those agents. Therefore, the separation distance S_d is also considered when adjusting the sensor range as it controls the influence of the local interactions.

Theoretically, the minimum number of frames required for an agent to reach a target is when it detects no objects during its path to the target. This can be done by setting the sensor range to zero which implies switching off the flocking system. Consequently, the number of frames times the step length an agent moves each frame exactly equals the direct distance to the target. However, the fastest route implies that agents move as a set of individuals rather than as team members. This also implies that agents do not interact locally with the surroundings in the environment; they do not align, cohere, or avoid colliding with other objects.

On the other hand, using a large sensor range implies increasing the influence of the flocking system. This in turn increases the number of interactions as each agent moves towards the nearest agent to align with, as well as towards the team centroid to keep itself bound to the team. This implies that the number of frames or movements, towards the target increases and accordingly, an agent takes a slower route towards the target.

The effect of the minimum distance allowed between objects in the environment, that is the separation distances S_d, is also considered in the experiment. The visual tests have shown that with S_d less than $7 units$ [1] an agent is happy to slightly bump into things, as

[1] A unit is used with respect to the environment dimensions, in other words, if the dimensions of the

this distance is less than the agent's dimensions (radius, width). In reality, the separation distance must allow for an object to turn safely without hitting the detected object. In contrast, increasing the S_d increases the time for an agent to complete the task. This is partially because with a large separation distance an agent may not be able to go through the space between two closely detected objects or squeeze tightly round a corner.

For this, an experiment is designed to run the $LGC - Model$ with 20 agents. The user interactively chooses a start point to issue the team task command by giving the position of the specified target location. The agents are to move in a team to reach this target location. Within the experiment, the number of frames to complete the task is recorded as well as the number of arrivals.

The experiment monitored the number of frames as a function of sensor range over four rounds. In each round, a single sensor range is tested with four values for the separation distance. For example, the first round tests four combinations using a sensor range of 70 $units$. These combinations are: $(SR = 70, S_d = 7)$, $(SR = 70, S_d = 10)$, $(SR = 70, S_d = 15)$, $(SR = 70, S_d = 20)$. These combinations are specified during the visual assessment depending on the simulated robot's dimensions. The visual assessment resulted in determining the combinations one can use to test both the team performance and completion time. For each combination, five trials were performed and the numerical outputs were recorded. Table 5.2 shows the averages of the outputs after running the system using a different S_d for each sensor range. The results of running the system using different population sizes (5 and 45 agents) are presented in appendix C within tables C.1 and C.2.

environment is 1700 $units$ $\times$ 800 $units$ the separation distance is 7 $units$

Table 5.2: Completion time in terms of the sensor range using different S_d values for 20 agents.

Sensor Range	Completion Time (Frames)			
(Units)	$Sd = 7$	$Sd = 10$	$Sd = 15$	$Sd = 20$
70	11248	12259	13319	14450
60	9698	10782	11898	12950
50	7215	8421	9450	10512
40	5446	6581	7680	8760

Figure 5.6 shows that the number of frames to complete the specified task increases as the sensor range increases. The increase in the number of frames, is an indicator of the increase in the distance, and implies that an agent was highly influenced by the flocking rules which prevents the agent, to a certain degree from taking a direct route towards the target, see figure see figures 5.7.

Increasing the sensor range results in an increase of the perception zones for the flocking system; i.e. an agent may see more agents, and therefore, the number of interactions increases, see figure 5.8. This implies more steps towards the target represented by more number of frames, see figure 5.9.

To summarise, choosing a lower value for the sensor range leads to reducing the influence of the flocking system. This in turn yields many individualists rather than team and group performance, each lives his own life for itself and does not try to co-operate with or follow others. Nevertheless, it is essential to maintain the influence of the flocking system by setting the sensor range that helps keep the essence of the group co-operation in the form of the team movements.

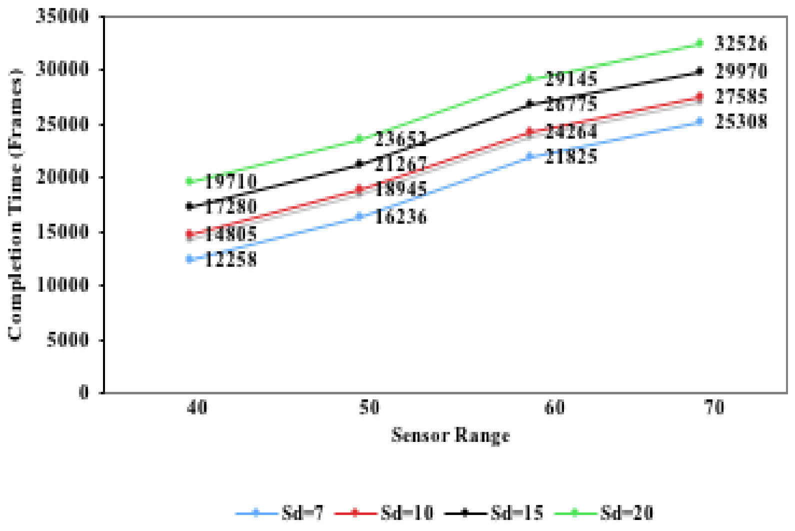


Figure 5.6: Number of frames as a function of sensor range.

5.2.3 Controlling the Interaction Weights of the Agent's Subsystems

This section tests, with the aid of visual evaluation, how the user can adjust the interaction weights to help accelerating the agents' progress? For this purpose, controlling the interaction weights is carried out by running the Local Communication Model ($LC-Model$), explained in section 4.3.1. As explained in section 3.3.1, the weights associated with the flocking rules are dynamically computed at each time interval τ depending on the rule's centroid and on the attitudes of each rule; these are the perception zone for this rule and the filtering strategy.

Originally, the flocking system is implemented here with the weights computed in a conventional way, see table 5.3. Since the collision avoidance weight is given precedence over other weights, as it is the most important interaction rule, the visual tests involved the

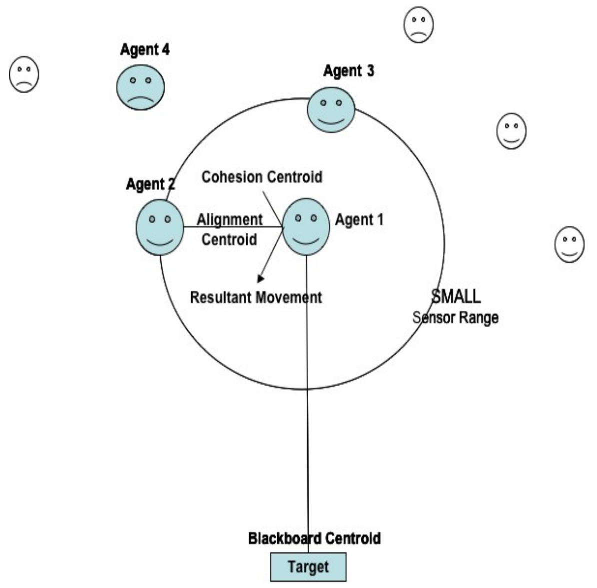


Figure 5.7: The smaller sensor range reduces the influence of the flocking system.

influence of the cohesion weight ($w^{R_\beta}_{A_i}$) on the progress of the agents. The visual simulation has been useful at this stage in assessing and evaluating the extent to which varying the cohesion weight allows the agents in the same team to move as a unit. According to the visual evaluation, it was found that the cohesion weight slows the agents progress, due to the high influence of the cohesion force. In addition, it causes overcrowding in the surrounding area which is used as an indicator for examining the strength of this binding force.

Table 5.3: The values of set of interaction weights

Alignment $w^{R_\alpha}_{A_i}$		Cohesion $w^{R_\beta}_{A_i}$		Collision Avoidance $w^{R_\gamma}_{A_i}$	
$C^{R_\alpha}_{A_i} < S_d$	$C^{R_\alpha}_{A_i} > S_d$	$C^{R_\beta}_{A_i} < S_d$	$C^{R_\beta}_{A_i} > S_d$	$C^{R_\gamma}_{A_i} < S_d$	$C^{R_\gamma}_{A_i} > S_d$
$1/C^{R_\alpha}_{A_i}$	1	$1/C^{R_\beta}_{A_i}$	1	1	1

Consider the situation where an agent detects a large number of nearby agents, then

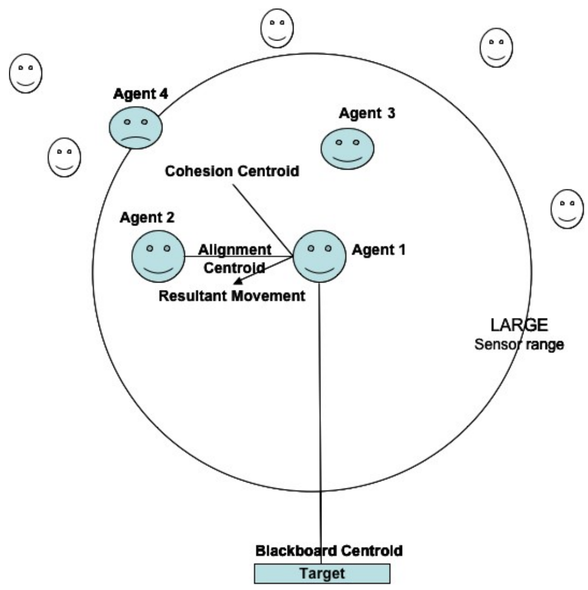


Figure 5.8: A larger sensor range increases the influence of the flocking system.

each of these agents modifies its velocity to move towards the cohesion centroid. If one or more of these agents detects a wall and at the same time some of the other agents within the avoidance zone, it may become trapped. In this trap situation, a neighbour of this agent (who may not detect the same objects) will be influenced by the trapped agent. In the same manner, the remaining agents will be influenced by the trapped agents as a result of a high cohesion weight. This can become worse if this set of agents is assigned a task to reach a specified target. Considering this scenario, the trapped agent continuously checks its capabilities of performing this task using the mechanism described in section 4.3.3. According to this mechanism, the trapped agent may discard his commitment regarding completing the task. The other agents who detect the trapped agent will be influenced by the trapped agent which can still significantly slow their progress. This leads to a longer expected completion time, or even prevents the influenced agents from completing the task.

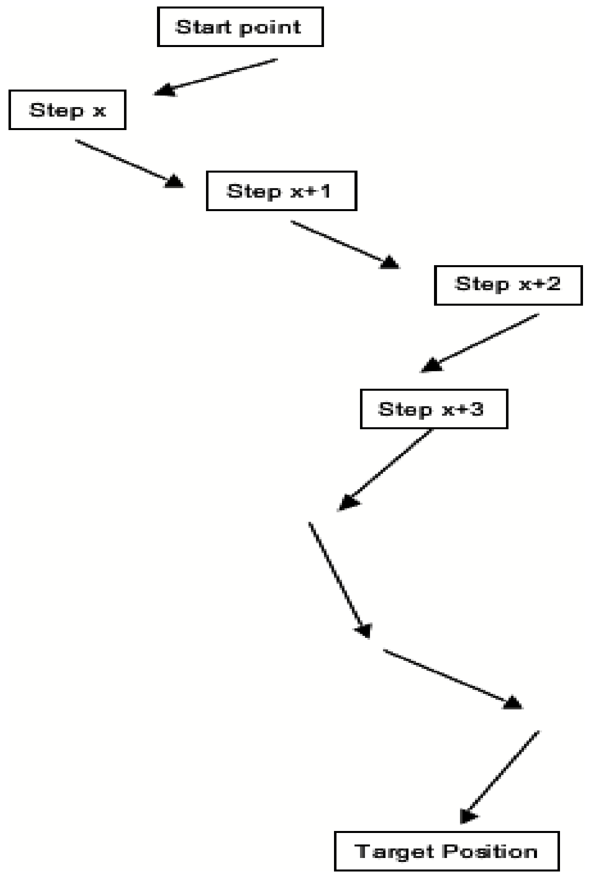


Figure 5.9: The number of movements towards the target increase as a result of a higher influence of the flocking system.

In this respect, a main goal of analysing the interaction weights then is to adjust the cohesion weight in order to avoid these impacts of a high cohesion weights without loosing the benefits of the supportive role of this weight in the team performance. Therefore, the start point was to test the conventional implementation of the weights in flocking algorithms, and the values are shown in table 5.3, for the alignment $w_{A_i}^{R_\alpha}$ and cohesion weight $w_{A_i}^{R_\beta}$. For this implementation, the cohesion weight is computed as the inverse of the distance to the cohesion centroid ($C_{A_i}^{R_\beta}$) if the $C_{A_i}^{R_\beta}$ falls within the avoidance range, otherwise it set equal to one. This implies that the cohesion force is mostly inversely proportional to the distance to the centroid. The weight becomes bigger very quickly as the centroid position falls outside the avoidance range (S_d) whilst it does not become very small within the avoidance range.

In order to numerically assess the dominance of the cohesion weight in situations where the agents dos not detect any avoidance cases, the $LC - Model$ was used. Accordingly, these interaction weights are shown in figure 5.10. The bar graph shows the weights that control the strength of the interaction forces, according to the values shown in table 5.3, on an agent over the first 200 frames of the simulation. Points of high cohesion weight, in figure 5.10, implies that an agent will be highly influenced by the nearby agents, and via monitoring the trap problem can be observed.

Table 5.4: The interaction weights, modified.

Alignment $w_{A_i}^{R_\alpha}$		Cohesion $w_{A_i}^{R_\beta}$		Collision Avoidance $w_{A_i}^{R_\gamma}$	
$C_{A_i}^{R_\alpha} < S_d$	$C_{A_i}^{R_\alpha} > S_d$	$C_{A_i}^{R_\beta} < S_d$	$C_{A_i}^{R_\beta} > S_d$	$C_{A_i}^{R_\gamma} < S_d$	$C_{A_i}^{R_\gamma} > S_d$
$1/C_{A_i}^{R_\alpha}$	1	$1/(C_{A_i}^{R_\beta})^2$	$1/C_{A_i}^{R_\beta}$	1	1

To overcome the trap problem, $w_{A_i}^{R_\beta}$ is modified in the following manner. Within the cohesion range, $w_{A_i}^{R_\beta}$ is inversely proportional to the square of the distance to $C_{A_i}^{R_\beta}$ otherwise it is inversely proportional to the distance to $C_{A_i}^{R_\beta}$ elsewhere as in table 5.4. Figure 5.11 shows the effect of modifying the way the cohesion weight is computed in reducing the influence of the cohesion force as well as in maintaining the essence of team performance. This can reduce the expected completion time when assigning a task to these agents.

The Local Global Communication Model ($LGC - Model$) is used to examine the effi-

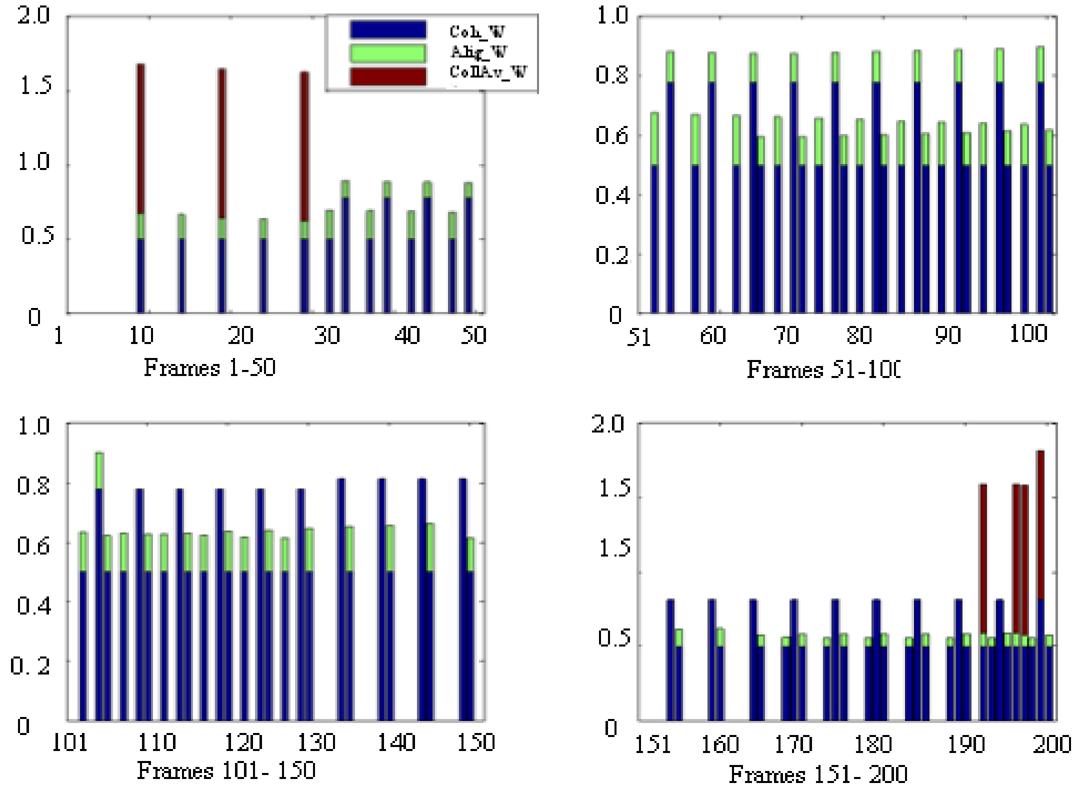


Figure 5.10: The interaction weights over the first 200 frames. The cohesion weight dominates the interaction weights whenever the avoidance weight is zero. The unmodified cohesion weight values are shown in table 5.3.

ciency of the modified weight of the cohesion rule in terms of completion time (in frames). In a comparison between the effect of the cohesion weight on the system performance using the values shown in tables 5.3 and 5.4, we switched the fourth rule on in order to record the completion time of a specified task for both cases. Table 5.5 shows the time elapsed until the first, 50%, and 100% arrivals, see figure 5.12. These values are the average values over five trials and all have a small standard deviation ($Dev <= 4$), (the row data values

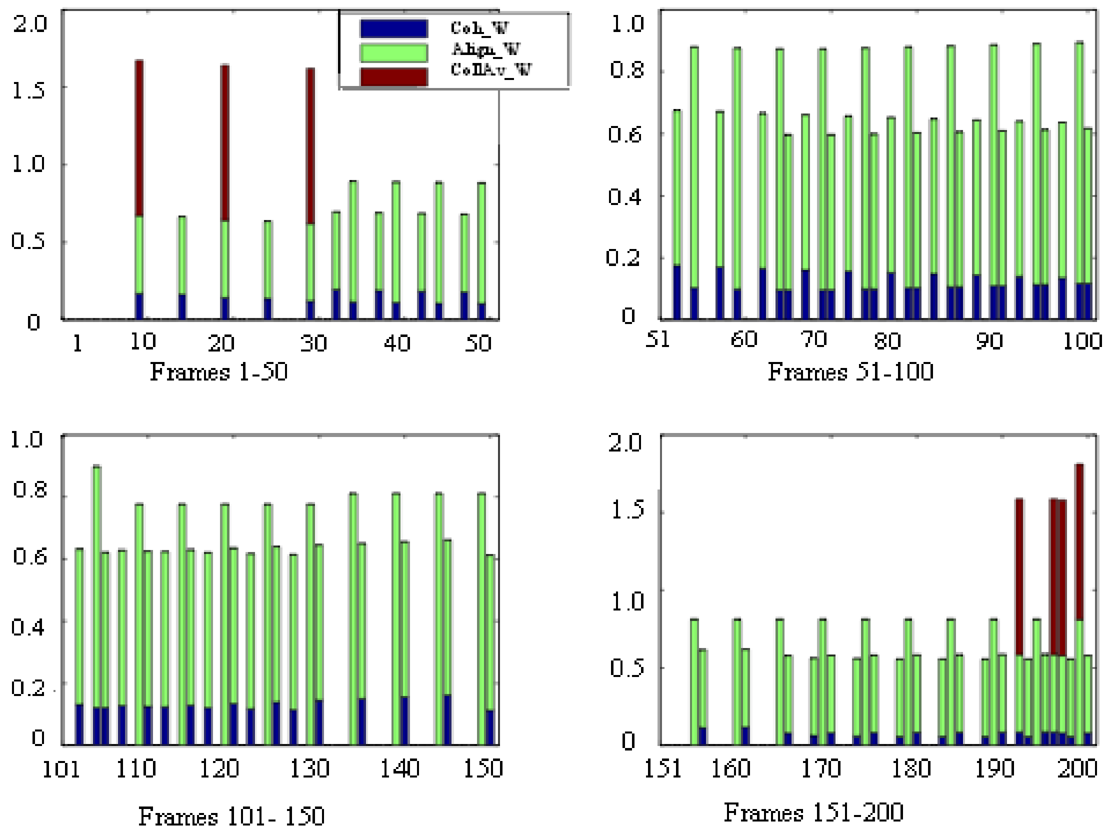


Figure 5.11: The interaction weights over the first 200 frames, with the cohesion weight modified according to the values shown in (table 5.4).

are presented in appendix C, table C.3 and table C.4).

Although, reducing the impact of the cohesion weight leads to minimizing the completion time, it also demonstrates a non-uniformness in the arrival rate for the set of agents. Investigating the uniform rate of arrivals is dealt with in detail in section 5.3 using the teaming technique and investigate the efficiency of the grouping technique by specifying various sizes of the teams to support a uniform rate of arrivals.

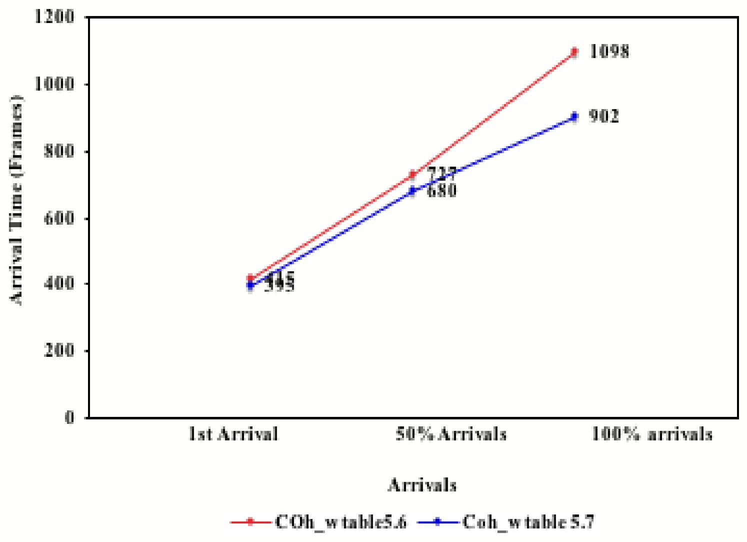


Figure 5.12: The completion time versus number of arrivals for a set of 20 agents, which shows the effect of reducing the cohesion weight, table 5.5

Table 5.5: The completion time for the first, 50%, 100% of arrivals for a set of 20 agents.

Arrivals	Unmodified Coh_w	Modified Coh_w
1st	415	395
50%	727	680
100%	1098	902

5.2.4 Detecting Locations of Complexity within the Environment

A novel aspect of the numerical analysis of the interaction weights, was the ability by analysing the cohesion weights to extract ideas about the structure of the environment. This implies how the *GBC* can draw a rough idea about the locations of complexity by comparing the cohesion weights with the results of analysing teams x, y positions in terms of their means and standard deviations.

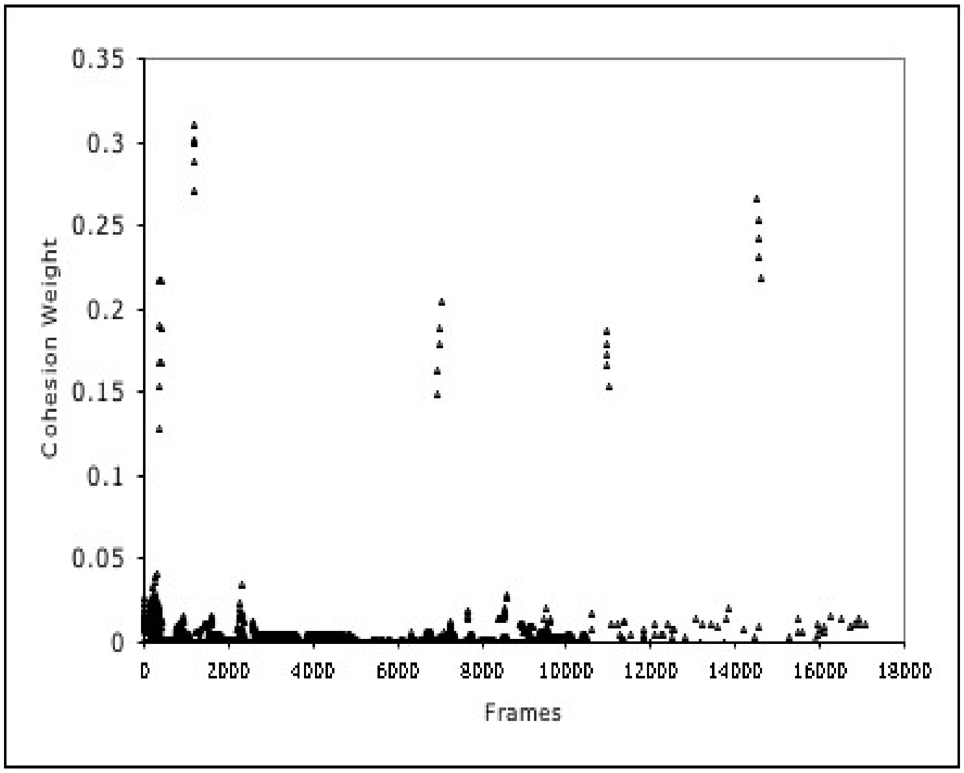


Figure 5.13: The distribution of the cohesion weights associated with the cohesion forces for each frame.

By considering that the communication protocol can be deployed as a communication with a higher level controller GBC, who could physically exist in a different location, the GBC can analyse the information sent by a set of agents so as to detect the locations of complexity in the environment. This is carried out by viewing the distribution of the cohesion weights while performing a specified task by running the $LGC - Model$. The experiment is designed to run the simulation by launching a set of five agents in the simulated environment shown in figure 4.7 from a start point, at the front of the screen, and assign them a task, to reach the target position shown in top left in the same figure. The numerical outcomes of the experiment consist of all the cohesion weights, and the (x, y) positions.

The graph shown in figure 5.13, shows the distribution of the cohesion weight along the frames. In this graph, low values of cohesion weight indicates a large distance to the cohesion centroid. A small cohesion weight may represent those agents which are moving away from each other in an attempt to avoid an obstacle. This also may imply that the detected agents are not close to each other and may be splitting up, depending on their positions from the obstacle and from each other, in order to avoid one or more obstacles. On the other hand, high values of the cohesion weights implies a reduced distance to the cohesion centroid, which indicates that the detected agents are close to each other. Therefore, the graph in figure 5.13 can be considered as a way to extract locations of difficulties encountered in the environment.

In addition, the test showed that the x, y positions can be fruitfully analysed together with the distribution of the cohesion weights to give a more explicit structure of the environment. Therefore, the x, y positions , sent by the agents to the GBC, are used to plot the deviations from the mean positions. During each frame, the mean positions of the agents and how far individually they are from the mean is calculated.

$$\bar{X} = \frac{\left(\sum_{i=1}^{N} X_i\right)}{N} \tag{5.1}$$

$$\bar{Y} = \frac{\left(\sum_{i=1}^{N} Y_i\right)}{N} \tag{5.2}$$

$$\Delta x_i = X_i - \bar{X} \tag{5.3}$$

$$\Delta y_i = Y_i - \bar{Y} \tag{5.4}$$

$$\Delta xy_i = \sqrt{\left(\Delta x_i^2 + \Delta y_i^2\right)} \tag{5.5}$$

$$\Delta \bar{x}y_i = \frac{\sum_{i=1}^{N} \Delta xy_i}{N} \tag{5.6}$$

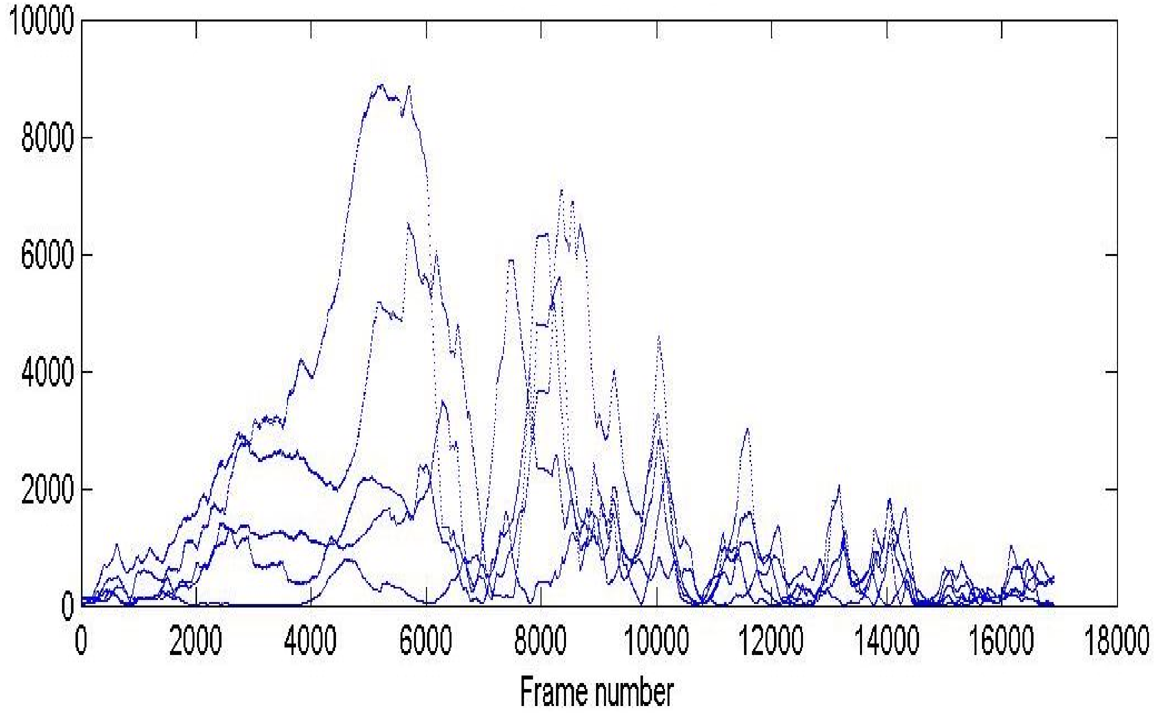


Figure 5.14: The individual deviations from the mean positions for the set of 5 agents.

The graph in figure 5.14 shows the deviation of the agents' positions from the mean, for the task used to analyse these cohesion weights above. The deviation follows five different routes and shows the times of the obstacles encountered, as high deviations, and the times of open areas as low deviations. For example, the time encountered and locations of the four rows of chairs shown in figure 4.7, can be extracted. This also can be useful in assessing the agents ability to autonomously organise themselves by maintaining their positions with respect to their neighbours. This implies that these agents were capable of maintaining their positions during each frame in order to act as a team on their way towards the target which supports the survivability aspect in the long term tasks. This level of autonomy is currently a high demand in robotic applications. This example has demonstrated the following issues:

- The set of agents are able to efficiently complete the task within a team so as to count the collision avoidance rule and social relations represented by the alignment and cohesion forces.
- The high level of autonomy these agents have whilst they are moving towards the target as they possess self-organizing and the position-maintaining in order to control their motion is considered efficient in controlling a large number of agents and supports the scalability objectives.

To conclude, the *GBC* can analyse the cohesion weights and can draw out an abstract picture of the environment by integrating the ideas from both graph 5.15 and graph 5.13, defining an explicit structure of the environment and the locations of complexity.

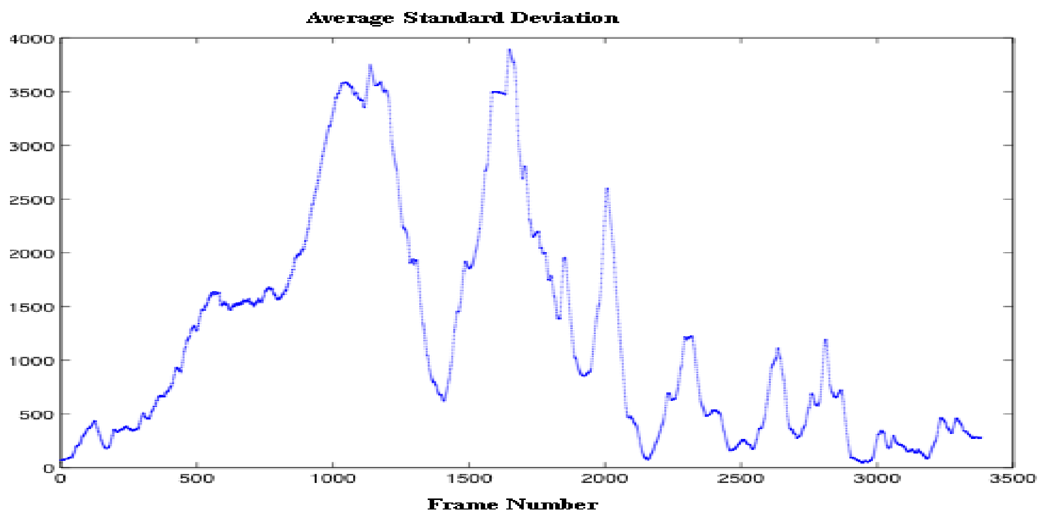


Figure 5.15: The average standard deviation for the same group, the set of 5 agents.

5.3 Grouping Technique: Optimum Team Size

The goal for this part was to study the effect of implementing grouping technique to optimise the uniformity of arrival rate. The test aimed at investigating the optimum team size when grouping a large number of agents into a set of small teams. For certain team sizes, there is not a uniform rate of arrivals at the target. A ratio (ρ) has been defined as the the ratio of team size to the number of teams. The values of ρ are computed from the possible combinations of the number of teams and the team size for a specified population size.

$$\rho = \frac{Team\ Size}{No.\ of\ Teams} \tag{5.7}$$

The experiment in this section is designed as follows:

- Specifying the number of agents for a specified task.

- Computing the possible combinations of team size and number of teams for this population size. For example, in order to study the emergent behaviour for a set of 48-agents, the possible combinations are shown in table 5.6.
- Running the $LGC - Model$ where the user issues the same task for the agents during each run. For each value of ρ, there are five trials.
- For each trial, the completion time for the first, 50%, and 100% of arrivals is recorded.

Table 5.6: The possible combinations, and resulting ratios ρ for a set of 48 agents.

Team Size	Number of Teams	Ratio ρ
1	48	0.0208
2	24	0.0833
3	16	0.1875
4	12	0.3333
6	8	0.7500
8	6	1.3333
12	4	3.0000
16	3	5.3333
24	2	12
48	1	48

The graph shown in figure 5.16 has two scales, the first scale is used to represent the number of frames to complete the task for the 50%, and 100% arrivals and the second scale is used to represent the number of frames for the first arrival with the values of the ratio ρ on the horizontal axis.

From this graph in figure 5.16 , one can see that the arrival time for the 50%, and 100% arrivals, as a number of frames, decreases as the number of teams increases. This is due

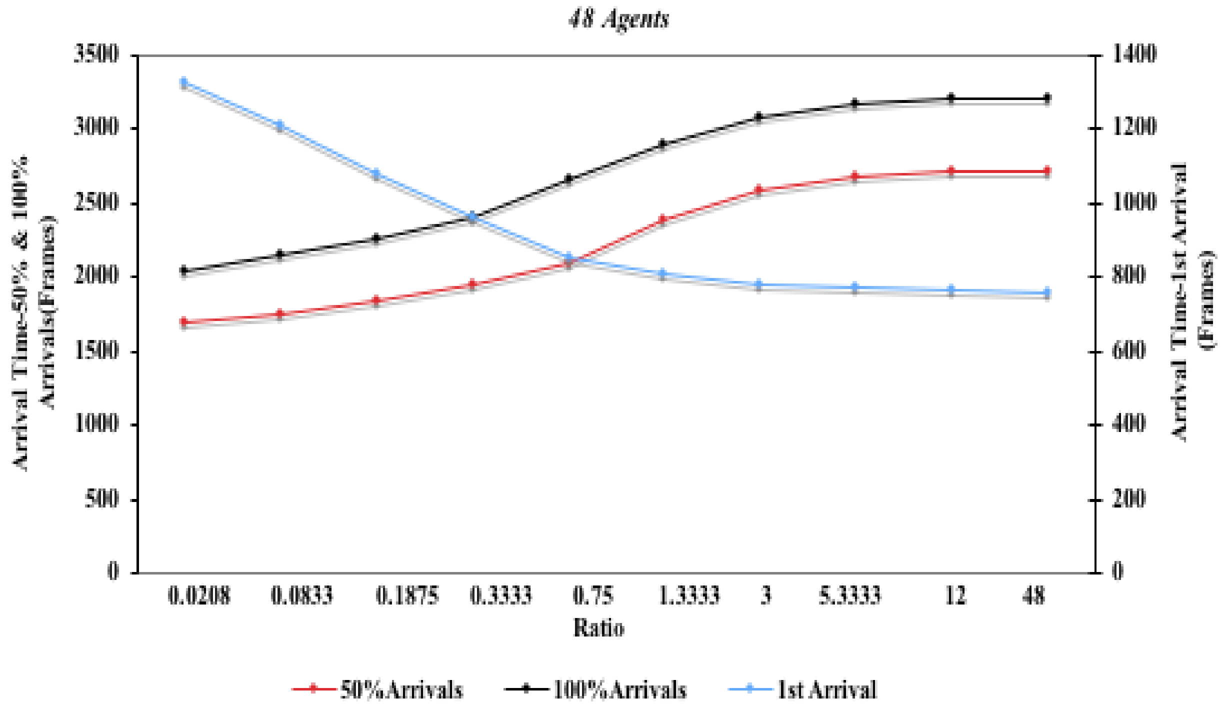


Figure 5.16: Variation in first arrival time, fifty per cent, and completion time vs ratio.

to the fact that a lower number of members in a team an agent needs to communicate and detect leads to a fewer social interactions. As the team size reaches one, the arrival time for the 100% agents is a minimum because the agents become a group of individuals moving towards the specified target with no local interactions except avoidance. This can lead to the agents arriving almost at the same time which results in overcrowding with respect to time at the target position. In other words, the smaller team sizes result in a less arrival time for the 50% of agents. The plots in figure 5.17 show the curve is negatively skewed as the ratio exceeds one which implies more clustering on arrival as they arrive almost in the

Table 5.7: The effect of varying ρ on the arrival rate, see table 5.6 for the ratio.

Ratio	Completion Time in (Frames) for the		
	1st Arrival	50%Arrivals	100%Arrivals
0.0208	1327	1700	2040
0.0833	1210	1750	2150
0.1875	1080	1850	2260
0.3333	960	1950	2400
0.75	853	2100	2660
1.3333	809	2380	2900
3	779	2580	3080
5.3333	772	2680	3170
12	765	2710	3200
48	761	2710	3200

same time.

On the other hand, a long arrival time for 50% of agents, represents a big team size, and implies that the first arrival may remain for long time waiting for the rest of the team to arrive at the target position. Although the first arrival can play a role in sending reports about the situation such that the rest of team can make use of this information, this might cause undesired situations for some practical implementations, for example a fire extinguishing task. In such a task, if the first agent arrived and started the task, this agent will be destroyed before a sufficient number of agents arrive and start the task. This is shown in the three bottom graphs of figure 5.17 , where a positively skewed distribution for a smaller number of teams is shown.

The plot that satisfies a normally shaped distribution that is most likely zero skewed was at $\rho = 0.7500$, the top right graph in figure 5.17. The $\rho = 0.7500$ implies 8 teams with

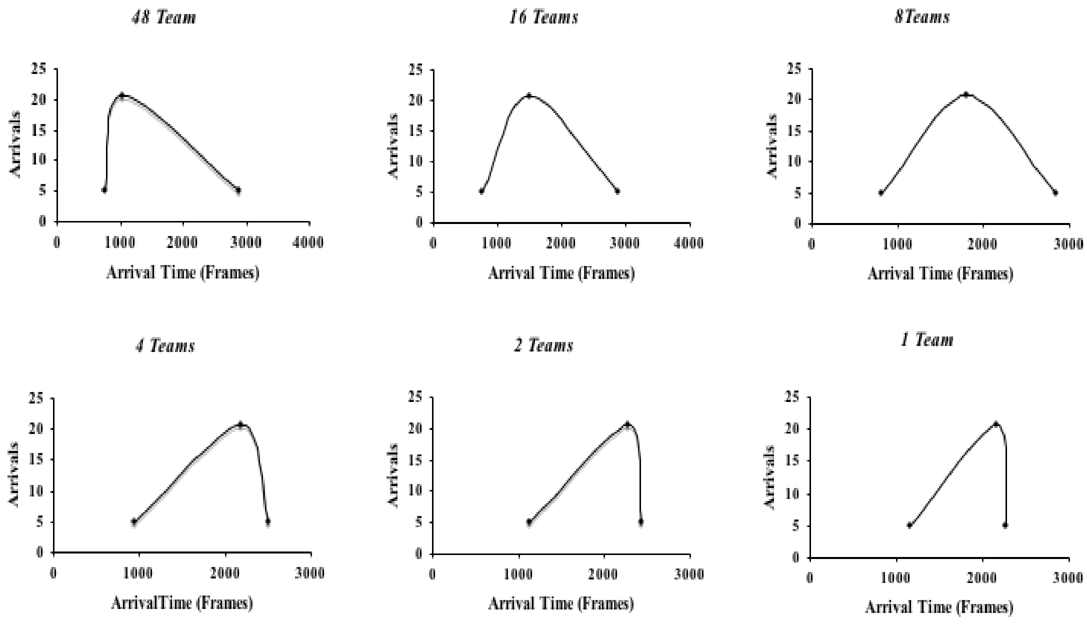


Figure 5.17: The uniformity of arrivals with respect to different ratios.

6 agents per team. This serves well in, for example, the scenario where a $CCTV$ camera is in operation on that site, it will see robots arrive at a constant rate to the location. This can be viewed as if there is a co-ordinator trying to co-ordinate those robots on arrival, it is easier for the co-ordinator to co-ordinate one by one on arrival rather than many arriving together. Thus agents in this situation can be considered to be self-coordinated.

The above experiment was repeated using a different population size (36 agents) for the purpose of comparing the results with those shown in figure 5.16 and figure 5.17. The possible combinations in table C.5 are used to compare the completion time for the first, 50%, 100% arrivals over these different ratios and similar results were obtained for the uniformity in the rate of arrival, see figure 5.19.

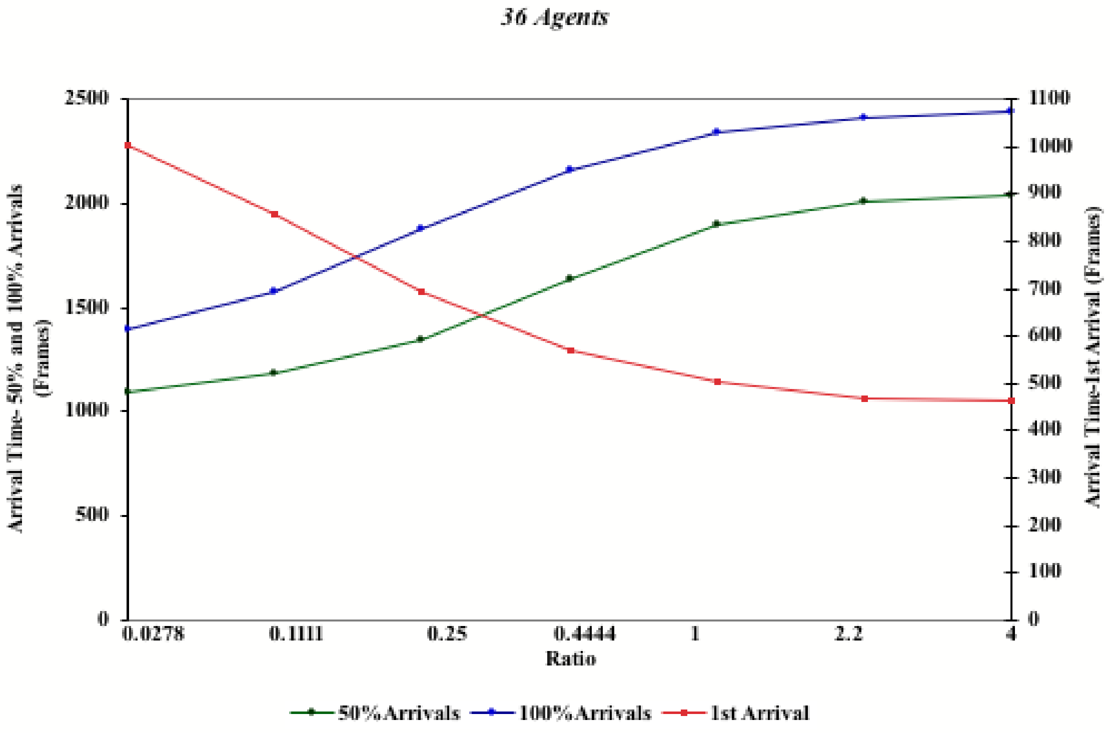


Figure 5.18: Optimum Team size, as a function of ratio for 36 agents.

This implies that teaming technique works with reasonable combination of team size and number of teams to lead to either a fast arrival or a reasonable arrival time for the whole team, and a uniformity in the arrival rate.

To conclude, the analysis in this section has shown that the proposed system can be used as toolkit to test different 'What If' situations. The real time simulation can support both the visual and numerical results without the need to use excessive time in testing for optimum team size for actual physical robots.

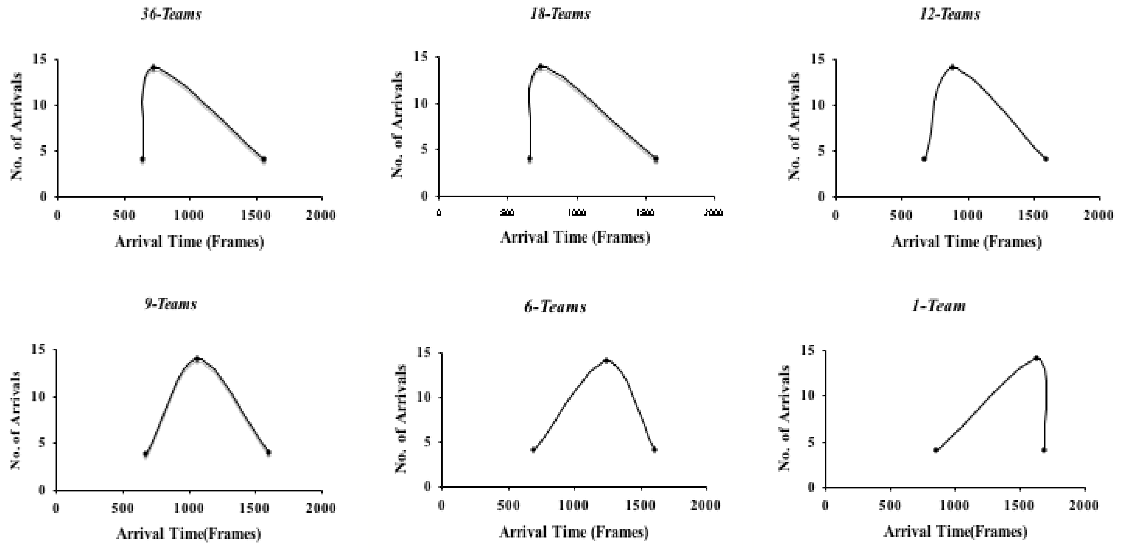


Figure 5.19: Optimum Team size, a uniform arrival rate with 9 teams of 36 agents.

5.4 Discussion

This chapter discussed the set of quantitative experiments used for the purposes of evaluating the system performance. A set of quantitative experiments was made aimed at analysing the influence of the flocking rules on the emergent behaviour. This required testing different combinations of sensor range, as the sensor links the agent to the environment, and minimum separation distance in order to adjust the level of the local interactions agents need to co-ordinate their movements. Therefore, the test aimed at finding the combination of sensor range and separation distance that satisfies: team performance and less completion time.

The experiments showed that using a small sensor range causes less influence from the flocking rules and at the same time reduces the number of movements an agent need to reach the target. This is due to the fact that reducing the influence of the flocking system causes less communication with the other local agents in the surrounding area and leads to individuals moving on their own without trying to co-operation with other agents. On the other hand, increasing the sensor range increases the completion time and leads to an overhead in communications as the number of detected agents increases. The experiments have shown that the most suitable sensor range that supports flocking behaviour is 40 *units* as it supports teamwork and reduces the completion time; i.e. the number of movements an agent need to complete a specified task expressed in the number of frames.

The influence of the flocking rules is also characterised by the weights associated with each rule. The numerical output of the experiments, with the aid of visual evaluations, also help in adjusting the cohesion weight, as it determines the strength of the cohesive force to be inversely proportional to the square of the cohesion centroid. The results of experiment after adjusting the cohesion weight, is compared to the results before adjustment. These results have shown that reducing the cohesion weight improves the system performance as it helped to speed up the agents' progress expressed by the reduction of the completion time.

Another issue that is considered important is the scalability of the system. The experiment has shown that the system scales up efficiently as it showed a linear relationship between the number of agents and the computational time.

Although teaming technique helps controlling the movements of a large group of agents and by issuing team tasks, therefore, in an experiment to test the self-coordinating ability of the agents, it has been shown that the agents can efficiently self co-ordinate their arrival time using the grouping technique. The grouping technique has been shown to be efficient if a suitable ratio is used, for example the 36 agents population size when grouped in nine teams is shown to be self-coordinated and can arrive at the target with an uniform rate.

Although, the presented work takes into account real-time considerations for monitoring purposes and timely issuing command, nevertheless internal computations must take priority. The effect of reducing the frame-rate is represented as the computational time in the experiment. The results has shown that the system scales up to fifty agents efficiently and this is represented by a linear increase in the computational time. When increasing the number of agents to more than fifty agents, the results has shown a non-linear increase in the computational time. This indicates that the graphical rendering of the agents and the simulated world is still the major bottleneck in the simulation. For a larger number of agents, more than 50 agents, the frame rate tangibly drops down due to the significant increase in the internal computations required. Finally, the incorporation of complex models increases the realism giving extra benefits, but adds complexity to the collision detection algorithms required to see if the sensor detects an object.

Chapter 6

The Design of a Co-operative Multi-Robots Team

This chapter shows the portability of our communication protocols from the simulation within a VE to a physical multiple robots system. Since it is likely that any simulator will require physical implementation, we show how this can be achieved. Therefore, this chapter discusses the operational specifications for how the proposed protocols would work in the real world on physical agents; i.e. robots. The group communication protocols specified in section 3.5 is considered for the case of multi-robots moving in an unknown dynamically changeable environment. This chapter is organised as follows:

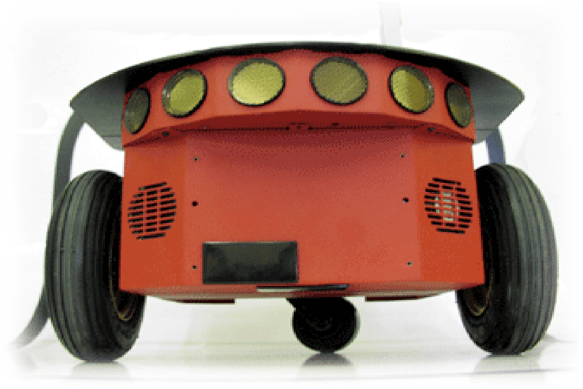
- The hardware description; the physical description of the robots used for this case study is presented in section 6.1.1
- The internet protocols considered, simple definitions, are presented in section 6.1.2

- The experiment descriptions and results are presented in section 6.2.

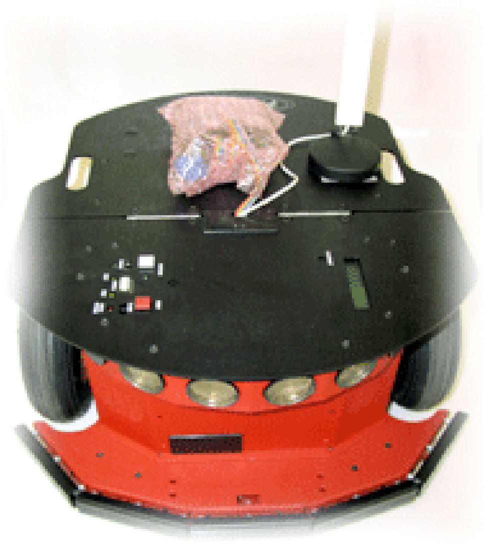
6.1 Operational Specifications

6.1.1 Hardware/Software Specifications

Pioneer robots, shown in figure 6.1a and 6.1b, were used in adopting the proposed communication model. The two robots are programmed with the *C* language under a linux environment and the package compiled using the *gcc*3 compiler.



(a) ***Chum.***



(b) ***Chuck.***

Figure 6.1: The robots with, although, different platforms, sonar sensors and little antenna for the wireless connection.

Each robot is 300 *mm* in width , 400 *mm* in length, and 400 *mm* in height. Each robot has a 20MHz Siemens 88C166 microprocessor with an integrated 32K ROM. The robots are also equipped with sonar sensors for collision avoidance. Each robot is provided with 8 sonar sensors. Although Chum and Chuck are of different shapes, they have a common platform as they both possess a motherboard that integrates intelligent motion control, and motor drivers with a small d.c. motor. They can be seen as a linux box on wheels that embeds TCP/IP networking capabilities. For the purpose of message exchange during communications, the robots and the user are networked according to the local area network protocol LAN. The robots and the user are wirelessly connected to this local area network LAN as the wireless connection, an ethernet connection, adds flexibility to the moving robots. The robots are accessed via the wireless connection would this enables us to log-in to any individual robot via the wireless connection, upload the code into a robot, and then start, stop or monitor that robot. Section D.1 presents the advantages of using linux-based robots.

To summarise, each robot must be provided with a linux-based operating system that meets the following fixed specifications:

- Support for wireless devices.
- Support for motor drivers interconnected with the operation system onboard.
- TCP/IP networking protocols.
- Associated programming tools (Sockets, Ports).

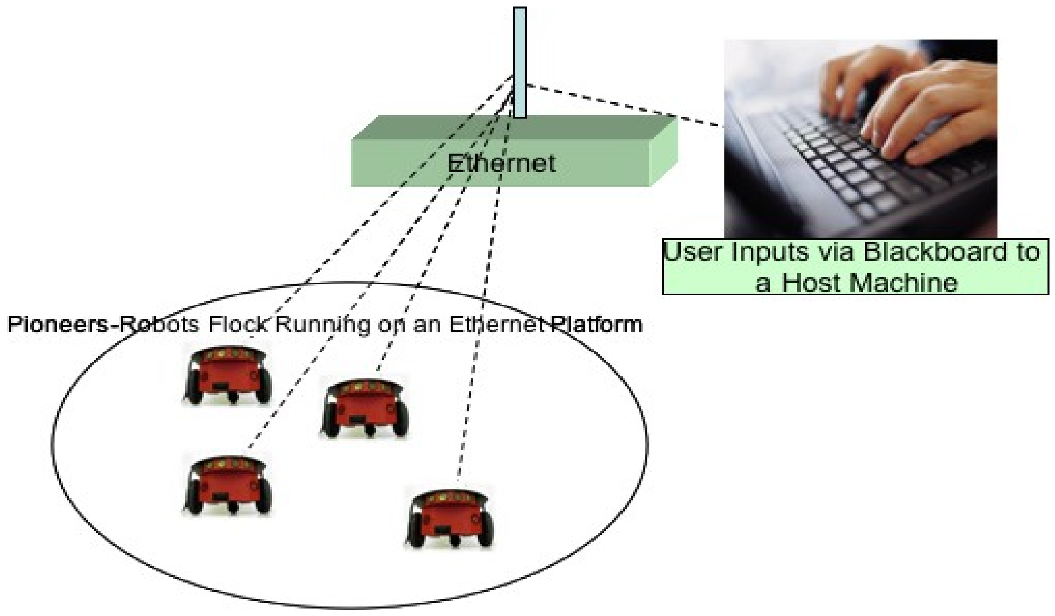

Ethernet
User Inputs via Blackboard to a Host Machine
Pioneers-Robots Flock Running on an Ethernet Platform

Figure 6.2: The robots recieve the user's commands as well as messages from other robots via the ethernet connection.

6.1.2 Network Protocols Specification

For the purpose of putting the proposed communication model, described in section 3.5, into practice with real robots, we had to consider some network protocols and definitions including the network settings specifications.

When an agent sends information and there is only one sender and one recipient then this is considered to be a unicast. A unicast is a one-to-one communication protocol in which a packet originates from a self-host, and it is destined to a unique recipient; i.e. another-host. For a multi robot system, the group interaction and communication require the establishment of a separate unicast connection with each of the recipients, or the use of a broadcast.

A broadcast in group communications is good if the sender does not specify the addressee(s) and wants all other agents including the *GBC* to receive this information. A broadcast refers to a one-to-many communication in such a way that a packet originates from a self-host, and is destined to all recipients within the same network.

Multicast is needed in our case study, where the *GBC* agent has information that should be transmitted to various, but usually not all, agents [19]. Such a group communication is presented in [20], as multicast refers to a one-to-many communication in such a way that a packet originates from a self-host, and the destination is multiple recipients within the same multicast address, the same team or group. In this context, an agent sends information to a certain special address via the external blackboard. Then, the addressees can pick, if they are interested in, the relevant information (positions and directions of near agents), and read them when they traverse the network. This is similar to broadcasting in that an agent sends only one broadcast packet which all the networked agents recognize and read. However, multicasting differs in that not all multicast packets are read and processed, but only those considered ”of interest”. The broadcast become similar to the multicast when all the agents in group are interested in all the packets sent by the other agents. In addition, implementing the multicast is needed in situations where the agents are divided into different groups, for example a search group and a rescue group. Agents may want to receive messages only from the group they belong to or messages sent only to the group they belong to. Multicast communication supports the external blackboard implementation as it allows extended multicast traffic to be available until interested recipients can read it.

The linux based operating system supports the broadcast protocols and unicast protocols but for the purposes of implementing the proposed protocol the network protocols for multicasting is the one that needs to be set up and thus the relevant settings are presented in appendix D.2.

The two robots and the user as a *GBC* are programmed as three clients and connected to a server hosted by the linux box via which they interact with each other, see section D.3.1.

6.2 Experimentation

Unlike the current implementations of the flocking robots, in our experiment the flocking communications are carried out across the network rather than the infrared whilst the collision detection is obtained by the sonar sensors. Therefore, we still have a dual perception system to deliver the inputs to the flocking system.

The collision detection system consists of three sets of sonar sensors; one set looking forward, one set looking to the front-left and the other set looking to the front-right. This sonar system returns only the range to the nearest obstacle from each sensor. Each robot can observe other objects falling in the front field of view (100 *degrees*) for the collision detection test.

In addition to the four communication rules, three are for the flocking system and the fourth is for the task level communications, the two robots have a default rule which is to continually move forward. The default rule helps in the case where any of the robots has miss-received a new message from an other robot.

The experiment aims at testing the following:

- Can the agents move within a group of two avoiding each other and the other obstacles.
- Is the user able to communicate with the robots whilst they are in motion.
- Are the robots able to broadcast/multicast a message to other robots within the same local network.

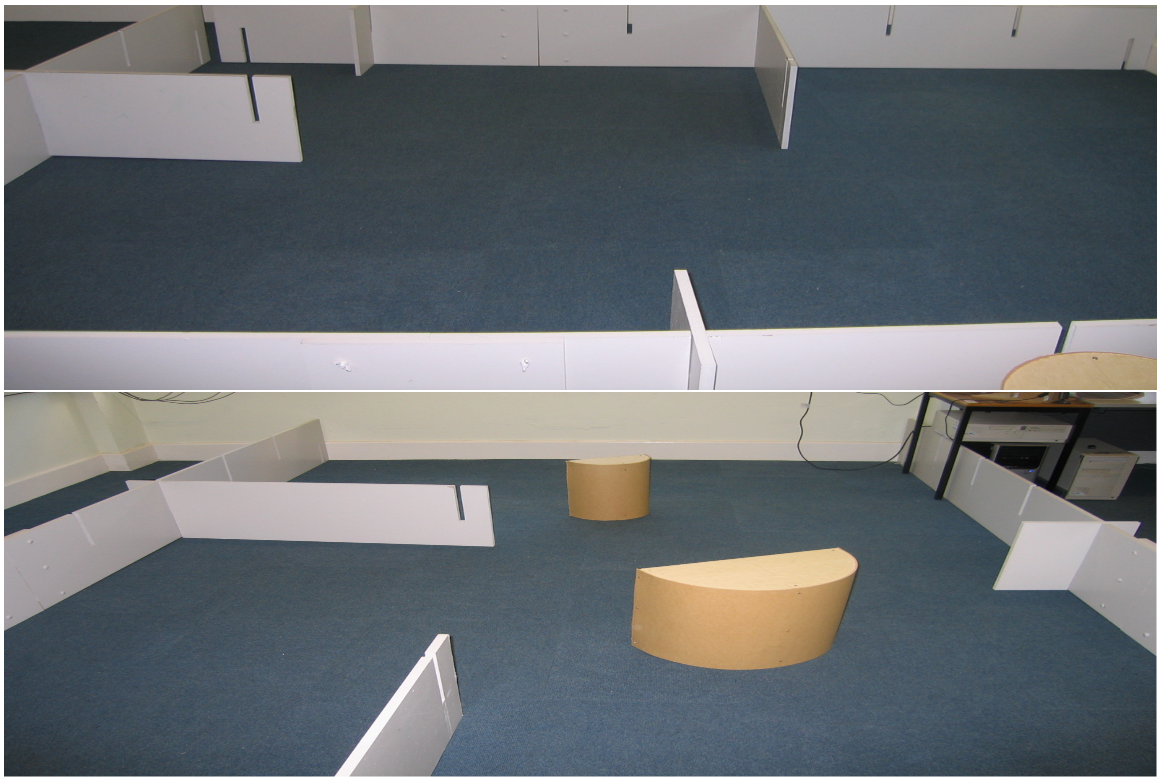


Figure 6.3: Experiment set up. Top, a maze; the environment where the robots operate. Bottom, the maze . with two removable obstacles.

The experiment is designed as follows: running the two robots, shown in figure 6.1 in a simple maze, shown in figure 6.3. The experiment starts by connecting the robots to the server and opening the ports. The user also connects via the server as a client to issue a team message for the robots to reach a point at the end of the maze. During the experiment, the user can add/remove some obstacles to test the robots' responses.

The camera-shots in figures 6.5, taken from the trials run in the robotic lab in De Montfort University, show how the two robots move as a unit trying to avoid the walls and aligning with each other. The experiment visual assessment has shown that:

- Each agent is able to send messages: multicast a message to all the agents, the other robot and the user, within the same local network. At this stage, the currently implemented phase, the agents multicast a message to all the agents within the same multicast address as we only have one multicast address.
- The message is composed of the sender robot ID, its belief expressed in the current position, and its intention expressed in the heading. The agent's beliefs and intentions form the contents of the sent messages, therefore the sent messages are of an informative type, as described in section 3.5. For example, in figure 6.5b, Chum's belief that there is a wall, and its intention, the changed heading, are sent in a message of the form:

  $CHUM: \ ALL \ pos(40, 50) \ \Theta[-45]$

  $CHUCK: \ ALL \ pos(90, 50) \ \Theta[-65]$

  The user's message is composed of the user's ID, then its recipients' IDs and the target position.

  $USR: \ ALL \ pos(100, 550)$

- Agents can communicate and maintain their positions in order to move in a team; e.g. Chuck and Chum try to join each other after they have passed the obstacle (figure 6.5d shows the effect of the cohesion rule).
- The robots also exhibit flexibility in their action selection during the movement in response to the dynamic nature of their environment, for example, the choice of the new heading with respect to the heading of the other moving robot and the position of the added obstacle, this is shown in figures 6.5c.

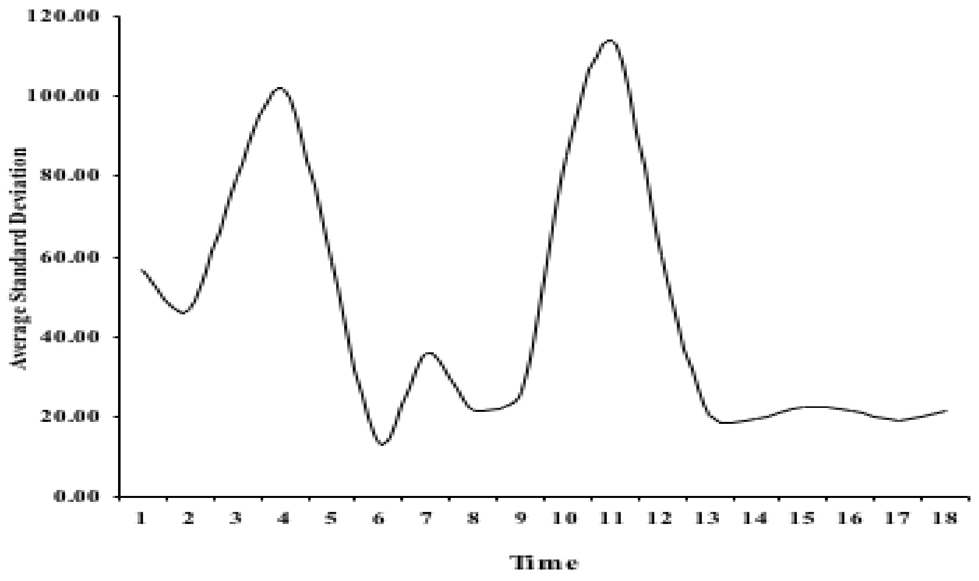


Figure 6.4: The average standard deviation, distance form the mean position for the two robots

Running this experiment with two robotos, Chum and Chuck, showed that the system, two robots, the user, and the host (server), in addition to the exchanged messages provide us with a small real world test bed. Because getting information sent and received within an ethernet operating on the robots allows us to also analyse the robots behaviour in the light of both real time responses to both the flocking goals and the user instructions, avoiding collisions, and moving in a unit.

The numerical outputs, x, y positions, from this experiment are analysed where the means of the robots positions are computed, as shown in figure 6.4. The plot shows the deviation of the robots' positions from the mean position versus an information update slot of time τ. The plot shows the effect of implementing the flocking behaviour on the robots positions whilst they are on physical movements. The small deviation also represents the two robots being close to the mean of their positions as they move closely to each other, whilst the high deviation represents the existence of an obstacle as the robots move away from each other in order to avoid this obstacle. Therefore, the high deviation expresses the obstacle avoidance whilst the low standard deviation expresses how close these robots are to each other.These results are very much similar to those presented in section 5.2.4 in figure 5.15.

6.3 Discussion

This chapter presented the pre-requirements for adopting the proposed MAS communication model to a real world case study. Therefore, the hardware specifications of the two robots used for our case study are presented and include the robots used in this work which feature an integrated computer that is running a Linux-based operating system with ethernet and wireless networking facilities. In this respect, group communication is carried out taking into account the messages exchanged via the ethernet connection. Since, the multi-agent systems can benefit from the possibility of broadcasting messages to a wide audience, the group interaction employs internet protocols such as broadcasting, unicasting and multicasting. According to these protocols, members in the group need to be able to send/receive messages in order to exchange information about the environment in the form of beliefs-desires-intentions.

The experiment run with the system consists of the two robots as two similar clients, the user as a different client type, and a host linux-box as the server. The results have shown that the robots were able to communicate with each other as well as with the user through the server built in to the host linux-box. The robots were able to maintain their positions according to the changes in the environment. In other words the new position of the other robot as well as the existence of obstacles.

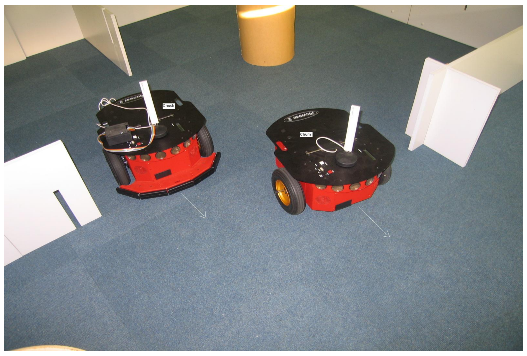
The results of implementing broadcasting within our MAS model has shown that there are many advantages in using broadcast communication in a multi-agent systems, e.g. sending multiple copies of the same message to each receiving agent within a group as the GBC issues the same message that carries for example a team task. Broadcast or multicast communication can save communication bandwidth by sending a single message destined for multiple receiving, mobile, agents. Secondly, multicast techniques can reduce a robots workload that leads to this robot not being able to do anything else but sending and receiving messages.

(a) ***Start positions and heading.***

(b) ***Chum detects a wall, and changes the heading. Chuck aligns with Chum.***

(c) ***Chuck and Chum are splitting to avoid an added obstacle.***

(d) ***Cohesion Rule.***


(e) ***Chuck is following Chum***

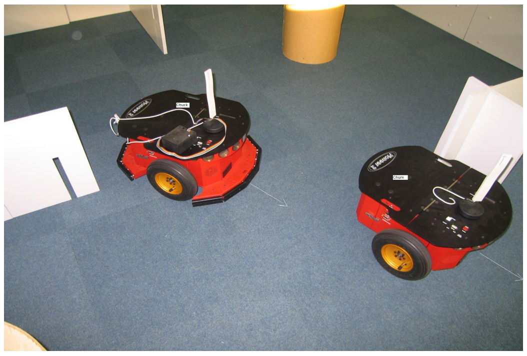

(f) ***Chuck and Chum moving as a unit.***

Figure 6.5: The two robots in operation.

Chapter 7

Conclusion

The work presented in this thesis introduced the tools to finalise a $PMAS$ interactive communication protocol through three phases. These phases, viewed as a pipeline, are: creating the full $PMAS$ communication model, building a VE that is used as a toolkit to test the model, and then implementing the model within real physical robots.

7.1 Interactive Communication within a $PMAS$ Model

The first major contribution is the interactive communication methodology for the task level interactions within a MAS. The proposed methodology integrates the flocking algorithm with the blackboard negotiation technique. In addition, two main enhancements were added to the common flocking algorithms, the first is filtering the inputs to the flocking system according to the requirements of each rule, and the second is use of heterogeneous weights and centroids in the flocking rules and for each agent by grouping the agents into teams. The enhanced flocking algorithm is used to minimise the extreme clustering of agents and support the team performance. Also, the flocking algorithm with an added randomness factor resulted in a more natural behaviour to the simulated flocks. The blackboard allowed for exchanging the messages presented in the form of the speech acts, that constitute the mental states and current intentions and goals.

7.2 A $HBDIB$ Architecture

The second major contribution is the novel architecture that is built for use with physical agents applications to meet the requirements of the proposed communication protocol. The proposed agent's architecture integrates the well-known Hybrid belief-Desire-Intention ($HBDI$) agent's architecture together with the Blackboard (BB) architecture into a Hybrid BDI-Blackboard $HBDIB$ architecture. The hybrid $HBDIB$ architecture combines the features of the perceive-act agent, with the blackboard architecture, with the a perceive-think-act agent of the $HBDI$ cognitive one in order to produce a higher level of interactivity for the perceive-think-act agent.

The hybrid $BDIB$ assumes sequences of mental states that cause actions as a result of cognitive dynamics that are adaptively coupled to the environmental dynamics. Accordingly, the communicating agent's situation has to include the detected team members' mental states, and their actions consist of speech acts. Agents must share at least the basic elements of their knowledge representational systems and be able to understand at least an outline of each other's cognitive dynamics.

7.3 The VE as a Testbed

The third major contribution of the work is the novel methodologies which are used to evaluate the communication protocol using the 3D simulation and visualisation tools. These tools provide a platform to easily develop different types of agent's architectures. In addition these tools allow for encoding different behavioural models. The implementation of the communication protocol within the VE has shown that the following advantages are obtained:

- Simulating a virtual perception system for each agent that is the sensor. The simulation tools allowed for visually assess the simulation of the sensor and the effect of changing the sensor range on the emergent behaviour.
- The ability to produce more realistic simulation: realism of the representation of the agents and environment, and also to simulate the rotating sensor that links the individual agents to the environment. Also, the realism of the representation of agents' actions.
- The 3D simulation and visualisation tools allowed for real-time user-interaction with agents.
- The ability to visualise a large number of agents at once by providing suitable tools such as multiple cameras and multiple viewpionts.
- Building a VE that imitates the real physical space gave an increased level of presence.

In addition, operating the agents in a virtual environment offers the advantage that the agent may be exposed to a variety of different tasks and surroundings without an inordinate amount of extra development time. Thus, the multi-agents system can be used as a testbed for higher-level tasks without the necessity of developing the extra hardware or transportation costs.

7.4 Verification

The set of experiments carried out through the evaluation process provided us with both visual and numerical analysis.

Visual analysis has been an effective tool in order to detect the bottlenecks of the system. For example, it helped in modifying the influence of the flocking system on the overall behaviours by allowing interaction and adjustment of both the weights and the sensor range.

The numerical analysis aimed at analysing the agents' positions in the experiments and has shown that the plots of the standard deviation of agents positions can help the user detect the locations of complexity in the environment. The experiments have also shown that the system scales up efficiently to 50 agents expressed in an effective linear relationship for the computational time with different population sizes. Also, the experiments have shown that the grouping technique, when used with suitable ratios, can improve the agents' self co-ordination and reduce the completion time as well as results in a uniform arrival rate.

7.5 Implementation in Real Mobile Robots

The model has been adopted to operate a linux-based operating system for the physical robots. The experiments have shown that the system can be successfully adopted to a real robots as the robots were able to maintain their positions with respect to the other team members.

In addition, the blackboard technique has shown that these robots, as clients connected to a static server can efficiently communicate with the user via the ground based-controller as another client through the same server.

7.6 Summary and Future Work

The work presented in this thesis has shown an efficient way to develop a communication protocol for a physical multi-agent system. The hardware implementation if it is to be extended helps in reducing the human presence in some real-life dangerous tasks such as cleaning toxic wastes, fire extinguishing or surface planet exploration. In addition, it can be used for wide search and rescue operations by grouping the robots into search teams and rescue teams. These teams can communicate to each others and also, based on the numerical analysis presented in chapter 5, the search team can draw a rough map of the regions of interest in the area. This map, partially created by the search team, can help the rescue team by reducing the time required to reach these regions of interest.

The results of the 3D simulations have shown some bottleneck within the system. These bottlenecks are caused by the complexity of updating and internally processing a large number of agents.

The work can be extended by developing a control system for a set of robots via the virtual environment. This can be done by extending the multicast protocol built in chapter 6 to include more than one multicast address which implies more than one team. This implies using more than one multicast address via the same server. At the same time, a virtual blackboard can be implemented using the 3D virtual environment that enables the user to interact with real robots through the virtual blackboard.

Bibliography

[1] G. Al-Hudhud, A. Ayesh, Martin Turner, and H. Istance. Simulation and Visualisation of a Scalable Real Time Multiple Robot System. In *Proceedings of the conference of Theory and Practice of Computer Graphics, TP.CG05*, University of Kent, Canterbury UK., June 2005. Eurographics Association.

[2] Ghada Al-Hudhud, Aladdin Ayesh, Howell Istance, and Martin Turner. Agents Negotiation & Communication within a Real Time Cooperative Multi-Agent System. In *Proceedings of the 5th International Conference on Recent Advances in Soft Computing*, pages 611–617, Nottingham, United Kingdom, December 16-18 2004. Nottingham Trent University. ISBN 1-84233-110-8.

[3] Ghada Al-Hudhud, Aladdin Ayesh, and Martin Turner. Speech Act and BlackBoard Negotiation Based Communication Protocol for real time Multi-agent Systems. In *Proceedings of the UK Workshop on Computational Intelligence UKCI-2004*, pages 112-120, Loughborough, United Kingdom, September 6-8 2004. Loughborough University. ISBN 1-874152-11-X.

[4] D. Andler. *Introduction aux sciences congnitives.* Gallimard, 1992.

[5] Antycip. *Vega Prime Training Manual, Version 1.2.* Distributors for MultiGen-Paradigm, Inc. A Computer Associates Company, 2003.

[6] R. Arthur, B. John, T. Micheal, and H. Jonathan. Co-ordination and Control of Multiple UAVs. In *AIAA Paper*, pages 45-88. Guidance Navigation and Control Conference, 2002.

[7] J. Austin. *How to do things with word.* Clarendon Press, 1962.

[8] A. Ayesh. Argumentative Agents-based Structure for Thinking-Learning. In *IASTED International Conference Artificial Intelligence and Applications (AIA 2001)*, Merbella, Spain, 2001.

[9] M. Barbuceanu and M. Fox. COOL - A Language for Describing Coordination in Multi Agent Systems. Enterprise Integration Laboratory, University of Toronto, *http : //www.cs.umbc.edu/kqml/papers/kool.ps.*

[10] M. Batalin and G Sukhatme. Coverage, Exploration and Deployment by a Mobile Robot and Communication Network. *In Telecommunication Systems, Special Issue on Wireless Sensor Networks*, 26(2), 2004.

[11] V. Becerra. Flocking Seven Dwarf Robots. Technical report, University of Reading, School of Systems Engineering, 2004. Cybernetic Intelligence Research Group, http://www.cirg.reading.ac.uk/robots/flocking.htm.

[12] M. Beer, M. d'Inverno, M. Luck, N. Jennings, C. Preist, and M. Schroeder. Negotiation in Multi-Agent Systems. *The workshop of the UK Special Interest Group on Multi-Agent Systems (UKMAS'98)*, 1998.

[13] C. Bererton, L. Navarro, R. Grabowski, C. Paredis, and P. Khosla. Millibots: Small Distributed Robots for Surveillance and Mapping. In *Government Microcircuit Applications Conference*, March 2000.

[14] G. Boella. Social Rationality and Cooperation. In *Proceedings of 2nd Asia-Pacific Conference on Intelligent Agent Technology.* World Scientific Publishing Co. Pte. Ltd. Singapore, 2001.

[15] J. Borenstein and Y. Koren. Optimal Path Algorithm For Autonomous Vehicles. In *Proceedings of the 18th CIRP Manufacturing Systems Seminar*. Stuttgart, 1986.

[16] J. Borenstein and Y. Koren. Real-time Obstacle Avoidance for Fast Mobile Robots. *IEEE Transactions on Systems*, 19(5), 1989.

[17] R. A. Brooks. Elephants Don't Play Chess. *Robotics and Autonomous Systems 6*, pages 3-15, 1990.

[18] B. Burmeister and K. Sundermeyer. Cooperative problem-solving guided by intentions and perception. *SIGOIS Bull.*, 13(3):10, 1992.

[19] P. Busetta, A. Dona, and M. Nori. Channeled Multicast for Group Communications - IRST Technical Report 0111-21. In *First International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS02)*, Bologna, Italy, July 2002.

[20] P. Busetta, M. Merzi, S. Rossi, and F. Legras. Intra-Role Coordination Using Group Communication: A Preliminary Report. In *International Workshop on Agent Communication Languages and Conversation Policies, ACL2003 (in conjunction with AAMAS03)*, pages 231–253, Melbourne, Australia, July 2003. Springer.

[21] O. Cairo, A. Aldeco, and M. Algorri. Virtual Museum's Assistant. In *Proceedings of 2nd Asia-Pacific Conference on Intelligent Agent Technology*. World Scientific Publishing Co. Pte. Ltd., 2001.

[22] D. Chapman. Planning for Connative Goals. *Artificial Intelligence*, 32(3):333-378, July 1987.

[23] Y. Chrysanthou, F. Tecchia, C. Loscos, and R. Conroy. Densely Populated Urban Environments, 2004. The Engineering and Physical Sciences Research Council, http://www.cs.ucl.ac.uk/research/vr/Projects/Crowds/.

[24] P. Cohen and H. Levesque. Intention is a choice with commitment. *Artificial Intelligence*, 42(3), 1990a.

[25] P. Cohen and H. Levesque. Performatives in a Rationality Based Speech Act Theory. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, pages 79-88, 1990b.

[26] P. Cohen and H. Levesque. Communicative Actions for Artificial Agents. In *Proceedings of the International Conference on Multi-Agent Systems*. AAAI Press, San Francisco, June 1995.

[27] P. Cohen, H. Levesque, and I. Smith. On Team Formation. *in G. Holmstrom-Hintikka and R. Tuomela (eds.) Contemporary Action Theory*, 2: Social Action, 1997.

[28] P.R. Cohen and H.J. Levesque. *Teamwork.* NOUS, 1991.

[29] B. Crowther and X. Rivier. Flocking of Autonomous Unmanned Air Vehicles. *Aeronautical Journal*, 107(1068), February 2003.

[30] W.J. Crowther. Rule-Based Guidance for flight vehicle flocking, Proc. *Instn. Mech. Engrs Part G (Journal of Aerospace Engineering*, 107(2):111-124, April 2004.

[31] F. Dignum and M. Greaves. Issues in Agent Communication. *Springer Verlag*, volume 1916, 2000.

[32] B. Donald. Agent Architectures. Computer Science Department, Dartmouth College, 1998, Web document available at http://www.cs.dartmouth.edu/b̃rd/Teaching/AI.

[33] E. Durfee, V. Lesser, and D. Corkill. Coherent Cooperation among Communicating Problem Solvers. *IEEE Transactions on Computers*, 36(11), November 1987.

[34] E. Durfee, V. Lesser, and D. Corkill. Trends in Cooperative Distributed Problem Solving. *IEEE Trans. on Knowledge and Data Engineering*, 1(1):63-83, 1989.

[35] E. Durfee and T. Montgomery. Coordination as Distributed Search in a Hierarchical Behaviour Space. *IEEE Trans. on Systems Man and Cybernetics*, 21:1363-1378, 1991.

[36] P. Eppstein. Finding the k Shortest Paths. *SIAM Journal of Computing*, 28:652–673, 1999.

[37] J. Ferber. *Multi-Agent System and Distributed Artificial Intelligence.* Addison-Wesley, 2002.

[38] Fipa Specifications, 2000. http://www.fipa.org.

[39] R. A. Freitas, T. J. Healy, and J. E. Long. Advanced Automation for Space Missions. *The Journal of the Astronautical Sciences*, xxx(1):1-11, January:March 1982.

[40] R. Fritzson, T. Finin, D. McKay, and R. McEntire. KQML - A Language and Protocol for Knowledge and Information Exchange. In *Proceedings of Distributed Artificial Intelligence Workshop, Seattle WA*, July 1994.

[41] L. Gasser and R. Hill. Coordinated Problem Solvers. *Annual Review of Computer Science*, 4, June 1990.

[42] F. Gentili and F. Martinelli. Optimal paths for robot group formations based on dynamic programming. *IEEE International Journal of Robotics and Automation*, 16(4):197- 206, 2001.

[43] M. Georgeff and F. Ingrand. Reactive Reasoning and Planning. In *The sixth National Conference on Artificial Intelligence,AAAI-87*, 1987.

[44] F. Giulietti, L. Pollini, and M. Innocenti. Autonomous formation flight. *IEEE Control Systems Magazine*, 20:34-44, December 2000.

[45] M. Gleizes, P. Glize, and S. Trouilhet. Etude des lois de la conversation entre agents autonomes. *Reveu internatonal de systemeque*, 8(1), 1994.

[46] R. Grabowski and P. Khosla. Localization Techniques for a Team of Small Robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01)*, volume 2, pages 1067- 1072, October 2001.

[47] R. Grabowski, E. Serment, and P. Khosla. An Army of Small Robots. *Scientific American, http://www.sciam.com*, November 2003.

[48] A. Hadadi. A hybrid architecture for multi-agent systems. In *Workshop on Cooperating Knowledge Based Systems (CKBS-93) University of Keele, UK*, 1993.

[49] A. G. Hernandez, A. El. Seghrouchni, and H. Soldano. BDI Multiagent Learning Based on First-Order Induction of Logical Decision Tree. In *Proceedings of 2nd Asia-Pacific Conference on Intelligent Agent Technology*. World Scientific Publishing Co. Pte. Ltd. Singapore, 2001.

[50] S. Ichikawa and F. Hara. Effects of Static and Dynamic Variety in the Character of Robots on Group Intelligence of Multi-robot System. In *Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems, DARS 2000*, pages 89–98, Knoxville, Tennessee, USA, October 2000. Springer.

[51] W. Jiao. Reasoning About Mutual Belief Among Multiple Cooperative Agents. In *Proceedings of 2nd Asia-Pacific Conference on Intelligent Agent Technology*. World Scientific Publishing Co. Pte. Ltd. Singapore, 2001.

[52] R. Jorna. *Knowledge Representation and Symbols in the Mind.* Tubingen: Stauffenburg Verlag, 1990.

[53] T. Kam, T. Gregory, Z. Wayne, and T. Ann. A Multiagent Operator Interface for Unmanned Air Vehicles. In *Proceedings of the 18th Digital Avionics Systems Conference*, pages 6.A.4.1–6.A.4.8, October 1999.

[54] P. Kearney, A. Sehmi, and R. Smith. Emergent behaviour in a multi-agent economics simulation, Cohn A G (Ed). In *Proceedings of the 11th International European Conference on Artificial Intelligence*, London, 1994. John Wiley.

[55] S. Kumar, M. Huber, and P. Cohen. Representing and Executing Protocols as Joint Actions. In *The First International Joint Conference on Autonomous Agents and Multi-Agent Systems(AAMAS-2002)*, July 2002.

[56] S. Kumar, M.J. Huber, D.R. McGee, P.R. Cohen, and H.J. Levesque. Semantics of Agent Communication Languages for Group Interaction. In *The seventeenth National Conference on Artificial Intelligence(AAAI-00) American Association for Artificial Intelligence*, pages 42-47, July 2000.

[57] K. Kuwabara, T. Ishida, and N. Osato. AgenTalk: Coordination Protocol Description for Multiagent Systems. In *Proceedings of the International Conference on Multi-Agent Systems (ICMAS '95)*, 1995.

[58] P. Langley. Cognitive architectures and the construction of intelligent agents. In *Proceedings of the AAAI-2004 Workshop on Intelligent Agent Architectures*, Stanford, CA, 2004.

[59] P. Langley, D. Choi, and D. Shapiro. A cognitive architecture for physical agents.

Technical report, Computational Learning Laboratory, CSLI, Stanford University, CA, 2004. www.isle.org/ langley/archs.html.

[60] J. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[61] D. Leevers, P. Gil, F.M. Lopes, J. Pereira, J. Castro, J. Gomes-Mota, M. Ribeiro, J. G. Gonalves, V. Sequeira, E. Wolfart, V. Dupourque, V. Santos, S. Butterfield, and D. Hogg. An Autonomous Sensor for 3D Reconstruction. In *3rd European Conference on Multimedia Applications, Services and Techniques (ECMAST98)*, Berlin, Germany, May 1998.

[62] N. Lesser and D. Corkill. The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks. *AI Magazine*, 4(3):15-33, 1983.

[63] H.J. Levesque, P.R. Cohen, and J.H.T. Nunes. On Acting Together. In *Proceedings of AAAI-90 Boston*, 1990.

[64] H. Li, W. Tang, and D. Simpson. Behavior Based Motion Simulation for Fire Evacuation Procedures. In *Conference Proceedings of Theory and Practice of Computer Graphics*. IEEE, 2004.

[65] F. Lin, D.H. Norrie, W. Shen, and R. Kremer. A Schema-based Approach to Specifying Conversation Policies. *Issues in Agent Communication*, 1916:193- 204, 2000. ISBN:3-540-41144-5.

[66] M. London. *Complexity and criticality in financial time series*. PhD. dissertation, De Montfort University, 2003.

[67] C. Loscos, D. Marchal, and A. Meyer. Intuitive Crowd Behaviour in Dense Urban Environments using Local Laws. In *Proceedings of the conference of Theory and Practice of Computer Graphics, TP.CG03*, University of Birmingham, Birmingham UK., June 2003. IEEE.

[68] C. Luo, S. Yang, and D. Stacey. Real Time Path Planning with Deadloock Avoidance of Multiple Cleaning Robots. In *IEEE International conference on Robotics and Automation*, September 2003. Tiwan,Thaibi.

[69] P. Marrow. Scalability in Multi-Agent Systems: The Diet Project, 2001. http://www.dfki.uni-kl.de:8080/DIET/public/index.html.

[70] M. Mataric. Designing Emergent Behaviours: from Local Interactions to Collective Intelligence. In *From Animals to Animates 2, Proceedings of the Second International Conference on Simulation of Adaptive 'Behaviour*, 1994.

[71] M. J. Mataric. Learning to behave socially. In *From Animals to Animates 3, Proceedings of the third International Conference on Simulation of Adaptive 'Behaviour*, Brighton,D. Cliff, P. Husbands,J. -A. Meyer and S. W. Wilson (Ed), 1992.

[72] K. Nagi. Modeling and Simulation of Cooperative Multi-Agents in Transactional Database Environments. In *Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS , 5th international conference on autonomous agents*, May 28June 01 2001. Montreal, Canada.

[73] L. Navarro, R. Grabowski, C.J Paredis, and P.K. Khosla. Millibots: The Development

of a Framework and Algorithms for a Distributed Heterogeneous Robot Team. *IEEE Robotics and Automation*, 9, December 2002.

[74] L. Navarro-Serment, R. Grabowski, C. Paredis, and P. Khosla. Millibots. *IEEE Robotics & Automation Magazine*, December 2002.

[75] H. Nishida and A. Takeda. Towards the Knowledgeable Community. In *Proceedings International Conference on Building and Sharing of Very-Large Scale Knowledge Bases '93 (KBKS '93)*, pages 157-166, 1993.

[76] H. Nwana, L. Lee, and N. Jennings. Coordination in software Agent System. Technical report, BT, Queen Marry and Westfeild College, 1994. Intelligent Systems Research Group, Applied Research and Technology Lab, BT Labs.

[77] P. Panzarasa and N. Jennings. Social Influence and the generation of joint mental attitudes in multi-agent systems, 2001. Department of Electronics and Computer Science, University of Southampton, UK, available at http://www.ecs.soton.ac.uk/ñrj/download-files/eurosim01.pdf.

[78] Research Program. Distributed Artificial Intelligence, 2000-2003. Web document available at: http://www.hds.utc.fr/b̃arthes/JPB_objectives.html.

[79] Group Project. Cognitive Architectures, 2000. Department of Electrical Engineering and Computer Science,University of Michigan, available at http://ai.eecs.umich.edu/cogarch2/index.html.

[80] A. Quintero, M. Eugenia, and S. Takahashi. Multi-Agent System Protocol Language

Specification, May 1997. Web document available at http://www.cs.umbc.edu/ cikm/iia/submitted/voewomg/yubarta.html.

[81] C. Reynolds. Flocks, Herds and schools: A distributed Behavioral Model. In *SIGGRAPH '87*, volume 21, pages 25- 34, July 1987.

[82] M. Rohrmeier. Telemanipulation of Robots via Internet Mittels VRML2.0 and Java, 1997. Institute for Robotics and System Dynamic, Technical University of Munchen.

[83] J. Schlecht. Mission Planning for Unmanned Air Vehicles Using Emergent Behavior Techniques, April 2001. Web Document available at http://www.cs.ndsu.nodak.edu/j̃oschlec/papers/uav_emergent.pdf.

[84] J. Searle. *Speech Acts.* Cambridge University Press, 1969.

[85] J. Searle. *Expression and Meaning.* Cambridge University Press, 1979.

[86] Y. Shoham. Agent-Oriented Programming. *Readings in Agents*, 1998.

[87] M. Singh. *Multiagent Systems, A Theoretical Framework for Intention, Know-How, and Communications. Springer Verlag*, LNAI, 799, 1994.

[88] M. Slater, A. Steed, and Yiorgos Chrysanthou. *Computer Graphics and Virtual Environments: from Realism to Real-Time.* Addison Wesley, 2002.

[89] I.A. Smith and P.R Cohen. Toward a Semantics for an Agent Communication Languages based on Speech Acts. In *Proceedings of the Annual Meeting of the American Association for Artificial Intelligence AAAI-96*, 1996.

[90] I.A. Smith, P.R. Cohen, J.M. Bradshow, M. Greaves, and H. Holmback. Designing Conversation Policies Using Joint Intention Theory. In *Proceedings of ICMAS-98 Paris, France*, pages 269-276. IEEE, 1990.

[91] M. Tambe. Agent Architectures for Flexible, Practical Teamwork. In *Proceedings of the 14th National Conference on Artificial Intelligence*, July 1997.

[92] W. Tang, T. Wan, and S. Patel. Real-Time Crowd Movement on large scale Terrains. In *Theory and Practice of Computer Graphics*. IEEE Computer Society, 3-5 June 2003. ISBN 0-7695-1942-3.

[93] F. Tecchia, C. Loscos, R. Conroy, and Y. Chrysanthou. Agent Behaviour Simulator (ABS): A Platform for Urban Behaviour Development. In *Conference Proceedings of Theory and Practice of Computer Graphics*. IEEE, 2003.

[94] F. Teccia and Y. Chrysanthou. Agent Behavior Simulator, 2001. Web Document, University College London, Department of Computer Science.

[95] Technical Report. Avatars and agents in immersive virtual environments, 2004. The Engineering and Physical Sciences Research Council, http://www.equator.ac.uk/index.php/articles/697.

[96] Technical Report. Network Robot Systems: Toward intelligent robotic systems integrated with environments. In *ICRA 2005 Workshop*. IEEE, 2005.

[97] M. Tirassa. Mental states in communication. In *Proceedings of the 2nd European Conference on Cognitive Science*, Manchester, UK, April 1997.

[98] B. Tsvetovatyy and M. Gini. Toward a Virtual Marketplace. In *Proceedings of The First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, 1996.

[99] R. Vincent, B. Horling, and V. Lesser. Experiences in Simulating Multi-Agent Systems Using TAEMS. *The Fourth International Conference on Multi-Agent Systems (ICMAS 2000)*, July 2000.

[100] T. Wan, H. Chen, and R. Earnshaw. Real Time Path Planning for Navigation in Unknown Environment. In *Theory and Practice of Computer Graphics*, Birmingham, UK., 3-5 June 2003. IEEE Computer Society. ISBN 0-7695-1942-3.

[101] T. Wan and W. Tang. Agent-based Real time Traffice Control Simulation for Urban Environment. *IEEE Transactions on Intelligent Transportation Systems*, 2004.

[102] T.R. Wan and W. Tang. An Intelligent Vehicle Model for 3D Visual Traffic Simulation. In *IEE International Conference on Visual Information Engineering, VIE 2003, Ideas, Applications, Experience*, 2003.

[103] T. Wanger and V. Lesser. Evolving Real-Time Local Agent Control for Large Scale MAS. In *Proceedings of 2nd Asia-Pacific Conference on Intelligent Agent Technology.* World Scientific Publishing Co. Pte. Ltd. Singapore, 2001.

[104] N. R. Watson, N. W. John, and W. J. Crowther. Simulation of Unmanned Air Vehicle Flocking. In *Proceedings of Theory and Practice of Computer Graphics*, pages 130–137, Birmingham, UK, 3-5 June 2003. IEEE Computer Society. ISBN 0-7695-1942-3.

[105] J. Wellner, S. Papendick, and W. Dilger. Scalability and The Evolution of Normative Behaviour. In *Proceedings of 2nd Asia-Pacific Conference on Intelligent Agent Technology*. World Scientific Publishing Co. Pte. Ltd. Singapore, 2001.

[106] K. Werkman. Knowledge-based model of negotiation using shareable perspectives. In *Proceedings of the 10th International Workshop on DAI*, Texas, 1990.

[107] M. Wooldridge. Agent-Based Computing. *Baltzer Journals*, September 1997.

[108] M. Wooldridge and N. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering*, 10:115-152, 1995.

[109] M. Wooldridge and N. Jennings. Software engineering with agents: Pitfalls and pratfalls. *IEEE Internet Computing*, 3:20- 27, 1999.

[110] W. Ye, R. Vaughyan, G. Sukhatme, J. Heidemann, D. Estrin, and M. Mataric. Evaluating Control Strategies for Wireless-Networked Robots Using an Integrated Robot and Network Simulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2941–2947, Seoul, Korea, May 2001. IEEE.

[111] R. Zlot and A. Stentz. Multirobot control using task abstraction in a market framework. In *Collaborative Technology Alliances Conference*, 2003.

[112] R. Zlot, A. Stentz, M. Dias, and S. Thayer. Multi-Robot Exploration Controlled By A Market Economy. In *IEEE International Conference on Robotics and Automation*, May 2002.

Appendix A

Conventional Implementation of Flocking Algorithms

A pilot study was conducted during the literature survey in order to practically evaluate the previous implementations of flocking algorithms. This system was developed and implemented in Java and the experiment ran the common rules of the flocking behavior for a set of point objects moving in a 2D arena.

A.1 Experiment

The experiment counted the interaction with the user was via the keyboard and the mouse. Agents interpret the position of the mouse as a specified task to reach this target point. The structure of the system can be described as follows: the Agent (entity) has been built that is aware of its mental states; positioning, belonging, turn- angle, and distance from other perceived agents. The Agent Manager acts as a resource manger in the design, through which agents can define social behavior and decision making such as agents turning, movements, conditions of addition, removal of agents, and calculation of target position. The Agent Platform acts as a second process, where agents are added, deleted, to be exposed to the environmental changes and emergent actions [3] and [2].

When studying the influence of the flocking rules on the emergent behaviour, a set of screenshots captured the behaviour of the agents. Each agent tries to match the speed and the direction of all other agents in the team, global alignment. How do these agents see each other? Imagine three agents flying close to each other in a simple structure. The area within which each agent can be seen is defined by a sphere whose radius is the maximum distance of visibility. A minimum separation distance is now defined as the radius of the collision detection zone. As the agents move in a group, an agent detects the other agents fall within its visibility sphere and tests whether they also fall within the collision detection zone then decides to follow or avoid. The three main rules applied in this prototype are adopted from the original flocking rules implemented in [81].

A.2 Recommendation and Summary

The representations of agents and the arena shown in figures included in section A do not support the real world problems with real physical sizes and spaces. There is still a need to visualise the agents with at least abstract shapes.

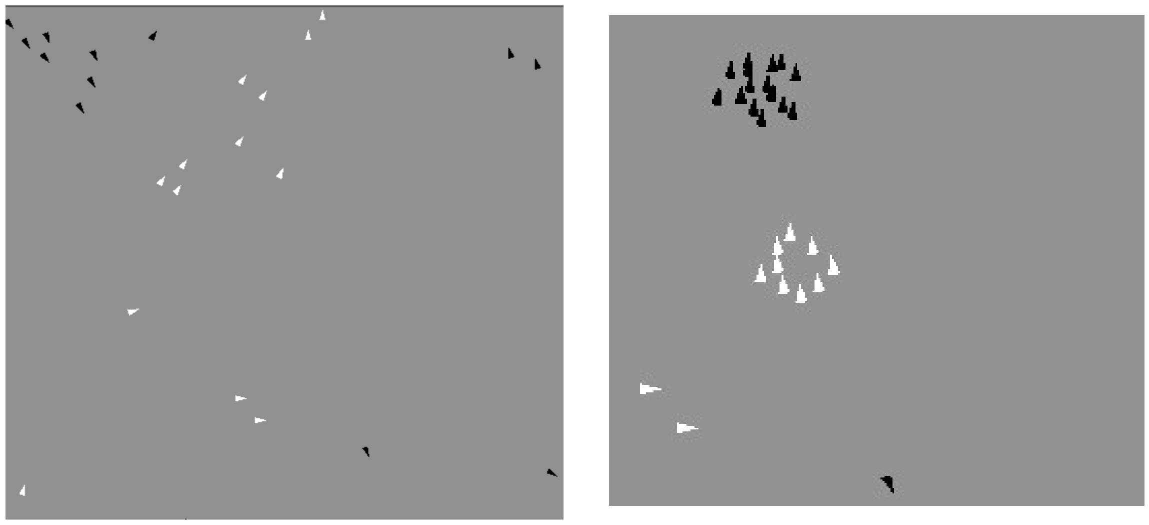


Figure A.1: Left, a randomly initiated set of agents. Right,cohesion force binds each team members together.

A main issue was spotted in those systems: that is locality queries are not satisfied as for each agent, and for all rules, all the agents within the team are included in the computations such that all the centroids in some sense define a global centroid. Therefore, our proposal was to allow each agent to filter the input data in order to test the detected objects only within a specified sensor range. In addition, each agent tests the identity of the other agents and if they are from the same team, it will find the offset vector to each agent then follow the used flocking rules.

Another issue also has been noticed was the way the system prioritises the actions. Within the old systems, the final decision is made by an agent always giving the priority to the collision avoidance rule, but if any object appears in the perception field, all the weights for the other rules will be reset until the agent avoids the object. According to the proposed protocol, all the weights are considered regardless of the existence of any obstacle.

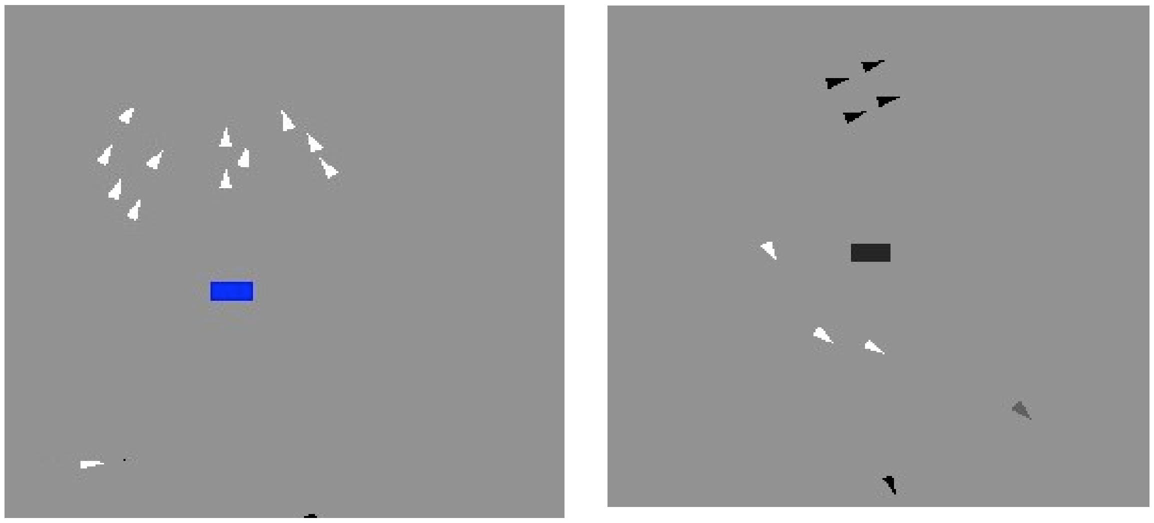


Figure A.2: Left, cohesion force binds the team members together after steering around the obstacle. Right, a screenshot shows both the alignment force and the collision avoidance rule.

Finally, many common systems implement the flocking algorithm as a motion control technique without assigning any higher task. The flocking algorithm is quite useful when trying to control the motion of hundreds of moving object to be seen as an organisation and not as a collection of individual moving objects. A major concern when developed the proposed system was to keep the user in the loop, so agents will receive some commands from a ground based controller for example.

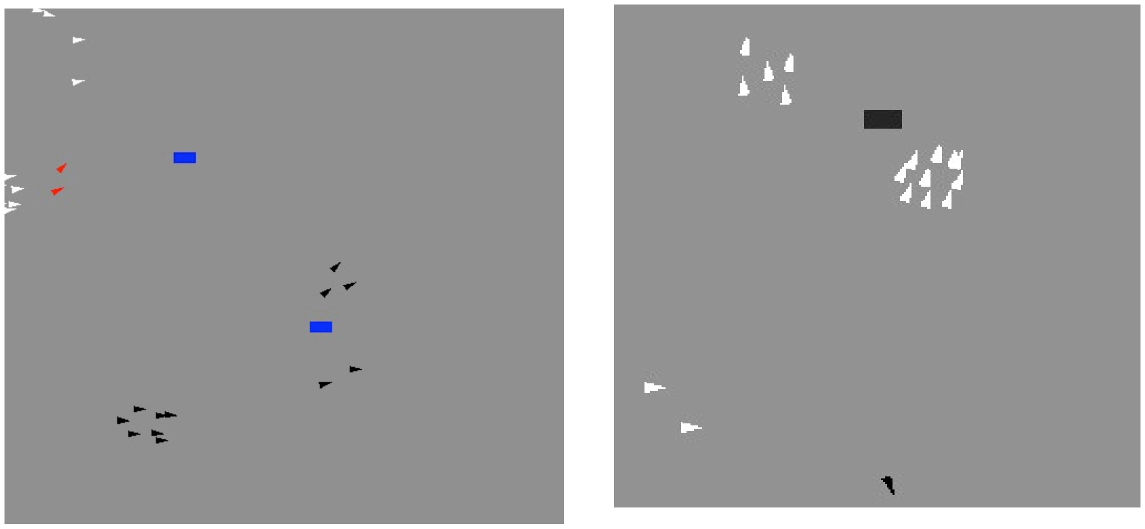


Figure A.3: Screenshots shows how the team members split into two teams to avoid colliding with the obstacle.Left, black-team members are split into two teams whilst avoiding the obstacle. Right, white team members are split into teams, note the two agents from the red team are trying to avoid hitting each other.

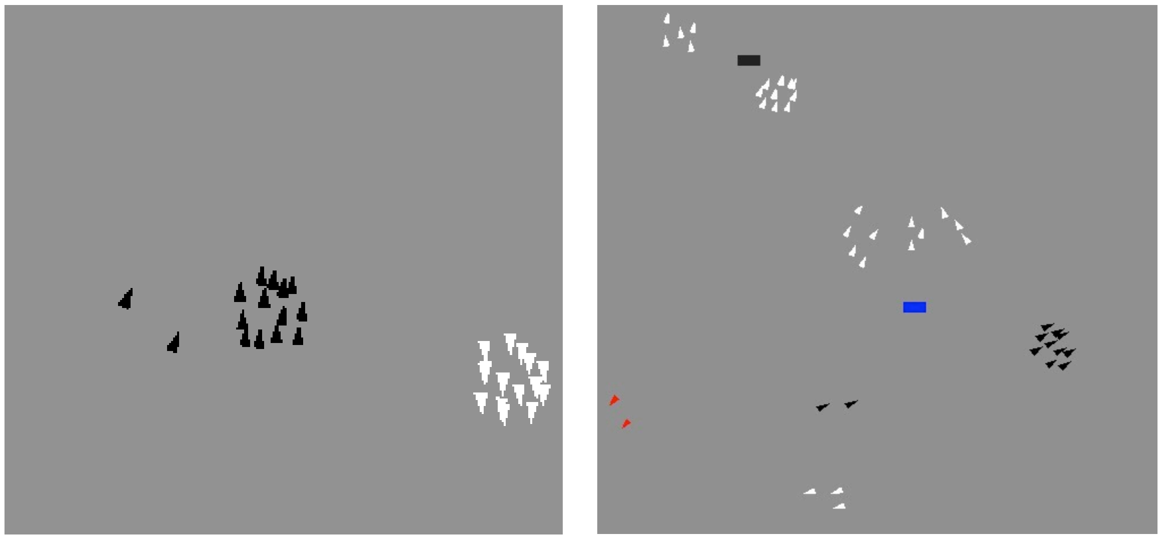


Figure A.4: Flocking Rules.

Appendix B

Developing a 3D Simulator

B.1 Software Development

The real-time software used can be described in [1] in two parts: (a) Representing the scene and the objects attached to it. This implies creating 3D models and has been done using a 3D modeling package (MultiGen Paradigm Creator). (b) Simulating the interaction between agents and rendering. The software environment VegaPrime is used for the real-time visual simulations including the scene description, interaction with, visualizing interaction between agents within the environment, and allows one to test the algorithm in real-time.

The user interface for Vega Prime is Lynx Prime where most of the set up is managed. The output of Lynx Prime is a VegaPrime Application Configuration File (ACF). *ACF* is the input for the application which is composed of the Vega Prime API and integrated

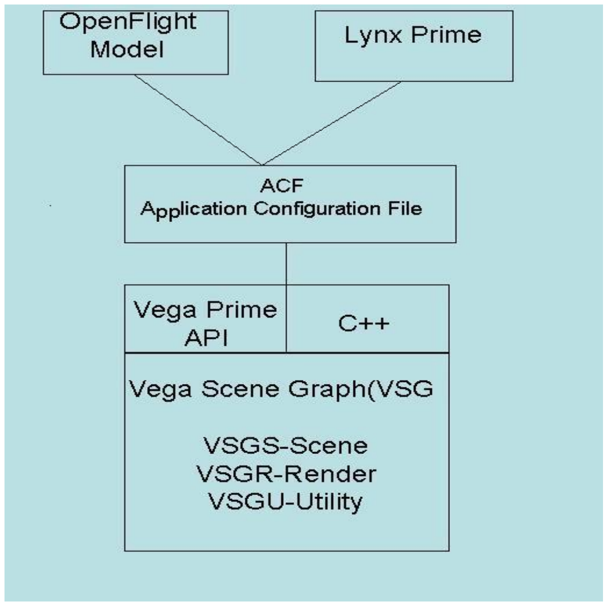


Figure B.1: Vega Prime System Architecture, [5].

C++ within which all the intelligence rules that an agent needs to move and interact with the environment are defined, B.1. The runtime control then includes:(a) defining the ACF, (b) configuring the ACF and the system, (c) executing the runtime loop which contains the user's methods, and (d) shutting down at the end of the application. In a VegaPrime application, after initializing the system the next step is to create a VegaPrime instance using the VegaPrime application class defined in (*vpApp.h*).

B.2 The Runtime Loop

The runtime loop consists of one function call: void vpApp::run(). The run() method executes the main simulation loop. The function will continuously call beginFrame() followed by endFrame() until the frame loop is terminated with breakFrameLoop(). Then the function will call unconfigure(). Table B.1 shows the run time loop and the following subsections describes the methods in the run time loop.

B.2.1 User Defined Application class derived from VegaPrime Application

```
    class myApp : public vpApp
{ public:
myApp() ; // constructor
m̃yApp() ; // destructor

void myUserDefinedPublicMethod();
void update (); //runtime
void run(){ update();}
virtual void onKeyInput(vrWindow::Key key, int mod)
configure():
};// end of Application class definition
```

B.2.2 User Defined Method

The user defined methods are:

- void moveObject(double step, vpObject *myObject) //User defined function to move object in ndirection

- void getsensordata()
- void updatesensordata()
- void postCheck(vpObject *myobjectPlane[],double heading[],int Mode[])
- User Interactive Inputs are being done via the keyboard: the method

  virtual void onKeyInput(vrWindow::Key key, int mod)

B.2.3 Configure Method: configure()

This method is defined in the vpApp class and handles the configuration process, reading ACF, initializing the kernel, pipeline content, window definition, and managing the relationships between the classes. This method is often overloaded by the user in a user-defined class derived from vpApp.

Table B.1: Vega Prime Run Time Loop

```
#include< vpApp.h >

int main(int argc,char * argv[])

{ // initialize vega prime

vp::initialize(argc, argv);

//create a vpApp instance

use the class have been created

myApp *app = new myApp;

// load acf file, assumes argv[1]is the acf file

app→define(argv[1]);

// configure the application

app→configure();

// runtime loop:

app→run();

// unref the app instance

app→unref();

// shutdown vega prime

vp::shutdown();

return 0;

}
```

Appendix C

Quantitative Assessments

This appendix presents the results of running the system under different settings, according to the set of quantitative analysis that was presented in chapter 5. This is included for the purposes of comparison with those results.

C.1 Sensor Range

This section presents the results for different trials in order to compare it with the results shown in section 5.2.2 and in table 5.2. Here, the experiment was to test the most suitable sensor range to be considered for further experimentation using two different population sizes (5 and 45 agents).

Table C.1 shows the results of running the system with 5 agents only whilst table C.2 shows the results of running the system with 45 agents, both starting from the same position

Table C.1: Completion time in terms of the sensor range and Sd values for 5 agents.

Sensor Range	Completion Time (Frames)			
(units)	$S_d = 7$	$S_d = 10$	$S_d = 15$	$S_d = 20$
70	2812	3065	3330	3614
60	2425	2696	2975	3238
50	1804	2105	2363	2628
40	1362	1645	1920	2190

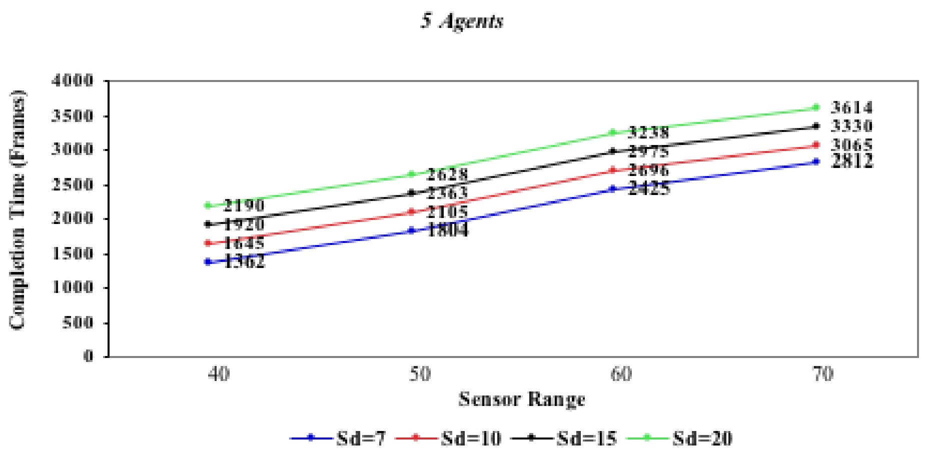


Figure C.1: The completition time as a function of sensor range for set of 5 agents.

and heading towards the same target. The figures shown in both tables show comparable results to those shown in table 5.2.

In comparison with the results presented in section 5.2.2, the sensor range influences the effect of the flocking system as the sensor data is the only input to the flocking system. Reducing the sensor range results in less interaction with other agents and therefore a reduced completion time. In contrast, increasing the sensor range leads to increases in the number of interactions with more objects being detected.

Table C.2: Completion time in terms of the sensor range and Sd values for 45 agents.

Sensor Range	Completion Time (Frames)			
(units)	$S_d = 7$	$S_d = 10$	$S_d = 15$	$S_d = 20$
70	25308	27585	29970	32526
60	21825	24264	26775	29145
50	16236	18945	21267	23652
40	12258	14805	17280	19710

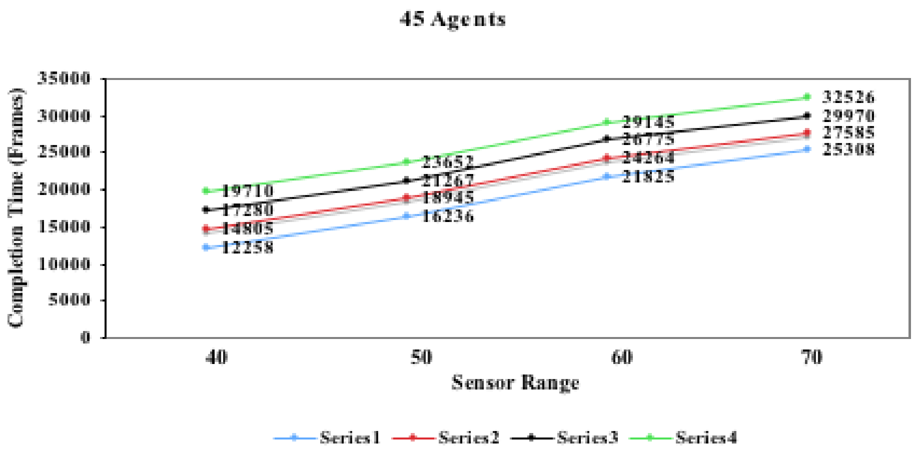


Figure C.2: The completition time as a function of sensor range for set of 45 agents.

C.2 Controlling the Interaction Weights

Table C.3 shows the result of running the system with unmodified cohesion weight values; that is the cohesion weight can be less than or equal one. Table C.4 shows the result of running the system with the modified cohesion weight ; how the cohesion weights are always less than one.

Table C.3: The completion time for the first, 50%,100% of arrivals for a set of 20 agents, $Coh_w <= 1$, see table 5.10.

Trial	Time in Frames		
	1st Arrival	50% Arrivals	100% Arrivals
1	419	730	1102
2	412	723	1094
3	409	732	1100
4	420	722	1097
5	416	729	1099
Average	415	727	1098
Average deviation	4	3.8	2.4

Table C.4: The completion time for the first, 50%,100% of arrivals for a set of 20 agents, $Coh_w < 1$, see table 5.11.

Trial	Time in Frames		
	1st Arrival	50% Arrivals	100% Arrivals
1	394	685	908
2	400	676	895
3	390	682	898
4	396	675	904
5	397	684	903
Average	395	680	902
Average Deviation	3.8	4	4

C.3 Optimum Team Size

For the purpose of comparing the results shown in section 5.3, for different population sizes, the system was run using a set of 36 agents. The possible combinations in table C.5 are used to compare the completion time for the first, 50%, 100% arrivals over these different ratios. The results shown in this table are similar to those presented in section 5.3. For that it has shown that the ratios less that one can supports the uniformity in the rate of arrival also for the set of 36 agents, see figure 5.19.

Table C.5: The effect of varying ρ on the completion time for first arrival, 50%, 100% arrivals for a set of 36 agents.

Teams Size	No. of Teams	ρ	Completion Time in (Frames) for the		
			1st Arrival	50%Arrivals	100%Arrivals
1	36	0.0278	1004	1090	1400
2	18	0.1111	857	1180	1580
3	12	0.25	694	1350	1880
4	9	0.4444	570	1640	2155
6	6	1	504	1900	2340
9	4	2.25	468	2010	2410
12	3	4	464	2040	2440

Appendix D

Network Multicasting Protocol

D.1 Why Use a Linux Based Operating System

The robots with a linux based- operating systems were chosen for the physical implementation because they are suitable and available in the Robotic Laboratory in De Montfort University. In addition, the linux-based operating systems meets all requirements to send, receive messages using TCP/IP (Transmission Control Protocol / Internet Protocol) within a network. TCP/IP is a suite of protocols which make up the basic framework for communication on the network. TCP is used to break the message into parts, known as packets and the IP is used to route the packets to the appropriate destination. The IP protocol manages the addressing of the packets and tells the router or gateway how and where to forward the packet to direct it to its proper destination.

Other protocols associated with the TCP/IP suite are UDP and multicast transmissions isused as a set of unicast packets using UDP. The transmission of packets through the network can be via cable or wirelessly. An example of wireless network is ethernet networks where the ethernet is a standard communications protocol embedded in software and hardware devices, intended for building a local area network LAN. In the ethernet wireless network, an ethernet card NIC is installed in each computer and is assigned a unique address and the wireless NICs use radio waves for two-way communication with a wireless switch or hub. An ethernet packet runs from each NIC to the central switch or hub that acts by receiving and directing packets of data across the LAN. In place of ethernet ports, wireless NICs, switches and hubs each feature a small antenna. Therefore, wireless networked robots have the ability to be more flexible to use.

The main pre-requirment was to adjust the IP network settings within which all the robot are connected wirelessly via an ethernet. Therefore, the two robots and the server, another linux box, are connected via an ethernet connection.

D.2 IP Network Settings

An Ethernet device normally retransmits each received IP multicast, broadcast or unknown unicast packet to all ports. Therefore, each robot, the user and the host which acts as a server knows the sockets and which ports to listen from. Each agent needs to send its information only once to a known address, multicasts its message, then the other agent picks the relevant messages and process the contents depending on their states of minds. Also, all agents need to be able to listen to all the self-messages sent to this address including self messages. All this requires a set of prespecified settings to be done in the network. Therefore, the set of underlying network settings is listed below:

- ($IP : multicasting$) protocol: Multicast is based on the concept of a group. Multicast is needed when configuring the kernel if a robot is to send and receive, i.e. multiple robots are registered to the same multicast IP address in order to send to and receive from the messages. This needs an $IPMulticast$ address to specify an arbitrary group of IP hosts that have joined the group and want to receive messages sent to this group. Hosts that are interested in receiving data flowing to a particular group must join the group hosts and must be a member of the group to receive the data stream. item Also when the Linux box is to act as a multicast router then it is important to enable multicast routing in the kernel by selecting $IP : forwarding/gatewaying$, $IP : multicastrouting$ to send multicast messages encapsulated within unicast ones.
- The data sent can be looped back to self-host by enabling the $IP-MULTICAST-LOOP$ protocol, i.e. loopback must be enabled for an agent that wants to *listen* to his own message.
- Finally, the sets of information are compressed into a (*packet*) along with a start of synchronisation code; a $packetID$. This $packet\ ID$ allows for the detection of recieved packets since if the ID of any received packet is not one of the team members, then it will not be considered.

According to the above settings, the main adaptive loop of the program is: read sonar system, collect / transmit packets from/to other robot(s), loop back. Then, an agent needs to update the weights, centroids, and correction angle when a complete packet of data is received) giving a higher priority to the collision avoidance. Finally, an agent decides on the next action, and evaluates the chosen action.

D.3 Programming

D.3.1 A Multirobot Chat Server

In order to allow the robot to select the server it needs to listen to or send to any packet, there is a *selectserver.c*, file that includes the following set of header files:

- #include $< stdio.h >$
- #include $< stdlib.h >$
- #include $< string.h >$
- #include $< unistd.h >$
- #include $< sys/types.h >$
- #include $< sys/socket.h >$
- #include $< netinet/in.h >$
- #include $< arpa/inet.h >$

The *main* function of *selectserver.c* defines the ports , we are listening on as each host need to be able to tune the two way communication channel by opening the port that allow it to send and receive via the socket through this port. Therefore, it includes first a set of declarations of:

- The master file descriptor list.
- The sockets addresses for the server as well as in the clients.
- The maximum file descriptor number
- The listening socket descriptor.
- The newly accept()ed socket descriptor
- The buffer for client data as an array of [256][3];
- The logical values to set the option for the socket (int yes =1 no =0) for setsockopt().

Then it runs through the existing connections looking for data to read, which includes: handling new connections, handling new data from clients, as we have different clients: user and multiple robots, by scanning the new data, and then constructing the replies by sending the data to everyone. This implies enabling the loopback so that each sender can listen to itself also.

D.3.2 Reading the User Inputs: User is a Client

In order to allow the server to read and receive the user's input via the keyboard, there is a $client - usr - KeyIn.cpp$ file that includes the following set of header files:

- #include < *stdio.h* >
- #include < *stdlib.h* >
- #include < *string.h* >
- #include < *sys/types.h* >
- #include < *netinet/in.h* >
- #include < *sys/socket.h* >
- #include "Aria.h"

The host needs to know about the connector's address information, the other hosts, and the network byte order. Take in the user's information (x, y) positions, it then write it as a message that is send as user's buffer to the server with an ID which is TAR to denote that this is the target position message form the user.

D.3.3 Robot's are Other Clients

The selfhost in each robot also needs to know about the connector's address information, the other hosts, and the network byte order. It reads the messages from the user and from the other robot, takes in the user's information (x, y) positions and the other robots information; the current position and the heading. After the robot decides on the next action, it sends the actual action to the motor driver whilst also sending the message as a client's buffer to the server with an ID which is CHM to denote that this is new information from Chum.

The distributed code that runs on each robot, need to be compiled onboard. The distributed code includes the following:

- Action class: This is where the robot gets the host name, socket's IP. Also, in this class the body of the action, a robot produces is defined to determine the type of interactions with the environment, GBC and the other robot. The robot exchanges messages as in this class; the type and the size of the message is defined. Finally, the robot's decision is also made in this class, see table D.2.
- The robot's main function: where the declarations of: the robot, the serial interconnection between the motherboard and the motor driver, the actions and the connected sonar devices. The main function first calls the Aria initialisation method, then calls the open connection method, sets the range of the sonar method, connects to the sensor devices, and then enables the motor driver method, and adds the action controller, then shuts down at the end of the connection, see table D.3.

Table D.1: User’s Main

```
#include < stdio.h >
#include < stdlib.h >
#include < string.h >
#include < sys/types.h >
#include < netinet/in.h >
#include < sys/socket.h >
#include ”Aria.h”
int main ( int argc, int argv[] ){
int sockfd;
char buf[100];
struct hostent *he;
struct sockaddr-in their-addr; // connector’s address information
he = gethostbyname(”localhost”);
//read form the keyboard
cout << ”Please,Key in the x-coor of the target: ”;
cin>> target-x; cout <<endl;
cout <<”Please,Key in the y-coor of the target: ”;
cin >>target-y; cout << endl;
//wirte message to a string msg
sprintf(buf,”TAR %f %f”,target-x,target-y);
//send this string to the server
if (send(sockfd, buf, strlen(buf), 0) == -1)perror(”send”);
printf(”The msg is being sent to the sever: %s”,msg); }
```

Table D.2: The Action Class

```
#include < errno.h >

#include < netdb.h >

#include < netinet/in.h >

#include < sys/types.h >

#include < sys/socket.h >

Action::Action(char * host,int sx,int sy) : ArAction(”Action”){

//1-Get the host name

//2- Check the socket address and the socket if connected

//3- Display a message says it is connected

//Body of the action

//——————First **Send msg*******————————

// Write the message into a string

////send this string to the server

//———Second **Recv target postion and other’s th x y**——

//1-Scan the recieved message and the check the message ID

//2-Write the contents in floats

//———————Third **Decidision Unit**———————

//1- Read the BB, i.e. the messages sent by the blackboard to the recipients in the network

//2- Compute the correction angles

//3- Mind the Collisions}
```

Table D.3: The Robot's Main Function

```
#include < Aria.h >

#include "Action.h"

#include < iostream >

int main ( int argc, char * argv[] ){

//Decalarations

ArRobot robot;//Declare the Robot

ArSerialConnection con; // Declare the serial interconnection in the robot

ArSonarDevice sonar;// Declare the sonar device;

Action driver(argv[1],atoi(argv[3]),atoi(argv[4])); //Declare the action instance

Aria::init();// Intitalise Aria

open the connection to the robot microcontroller/simulator

if ((ret = con.open()) != 0) // FOR ROBOTS

if ((ret = con.open("localhost",atoi(argv[2]))) != 0) { str = con.getOpenMessage(ret); }

robot.addRangeDevice(&sonar);//Set the range of the sonar

robot.setDeviceConnection(&con); //Connect the sensors devices

robot.comInt(ArCommands::ENABLE, 1); // enable motors

robot.addAction(&driver); //Add action controller

Aria::shutdown();//Shutdown

return 0;}
```

Appendix E

Publications

The following papers has been published from this work:

1. G. Al-Hudhud, M. Turner, and A. Ayesh. Speech act and blackboard negotiation based communication protocol for real time multi-agent systems. Presented at the 2004 UK Workshop on Computational Intelligence (UKCI-04), September, pages 112-120, 2004.
2. G. Al-Hudhud, A. Ayesh, M. Turner, and H. Istance. Agents negotiation & communication within a real time cooperative multi-agent system. Presented at the Fifth International Conference on Recent Advances in Soft Computing (RASC2004), December 2004.
3. G. Al-Hudhud, A. Ayesh, M. Turner, and H. Istance. Simulation and visualisation of a scalable rael time multiple robot system. In To be presented in the conference of Theory and Practice of Computer Graphics, TP.CG05. IEEE, 2005.

The hardcopies of the full papers are included in the following pages of the thesis.

Appendix F

Software System & Demos

The CD enclosed provides an electronic version of the thesis, and the relevant published papers. Additionally, there are three other folders: Java Demos, Video Captured Demos for the 3D simulator, and demos for the video captured for the twin-robots.