INTELLIGENT COLLISION DETECTION AND AVOIDANCE TECHNIQUES FOR AUTONOMOUS AGENTS

By

Fan Liu

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
AUCKLAND UNIVERSITY OF TECHNOLOGY
AUCKLAND, NEW ZEALAND
AUGUST 2014

© Copyright by Fan Liu, 2014

To Prof. Ajit Narayanan and
my parents Guoqing Liu and Yufen Zheng.

Table of Contents

List of Tables

List of Figures

Attestation of Authorship

"I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any other degree or diploma of a university or other institution of higher learning."

Auckland, New Zealand
August 2014

Fan Liu

Abstract

Collision is one of the main problems in distributed task cooperation involving multiple moving agents or robots. The collision avoidance problem arises when the environment is dynamic and to reach their destination agents need to use paths that conflict with other agents' paths on specific moves. Decentralized collision avoidance in these situations is more challenging than centralized collision avoidance since autonomous agents must manage their moves independently and may have only a limited capability (local view) to detect the potential risk of collision. Moreover, for true autonomy, there must be no communication between agents or with a central coordinator.

This thesis describes novel extensions to current approaches for dealing with collision avoidance and proposes a new dynamic rectangular roundabout ('rectabout') collision avoidance method based on human behaviour. The method uses Minimum Enclosing Rectangles (MERs) as potential roundabout carriers to form virtual rectabouts that allow each agent to re-plan its path autonomously and with no communication. This maneuver is calculated independently by each agent involved in a possible collision. The virtual rectabout lies in the intersecting and conflicting position of two agent routes. The approach does not depend on priority schemes and instead involves only local views.

MER in turn consists of two components: Minimal Predicted Distance (MPD) detection and MER rectabout collision avoidance algorithm. The MPD is a metric inspired by real human pedestrian collision avoidance behaviour. We use MPD to detect the possible collisions along agent paths and trajectories. The agents involved in conflict will compute a rectabout and re-plan a new velocity when MPD is below

the threshold.

Experimental simulations involving multi-agent systems indicate that the proposed approach ensures that all agents remain free of collision while attempting to follow their goal direction. The decentralized collision avoidance approach is also applied for WowWee Rovio mobile robots and provides both analytic and empirical evidence to show that the approach generates collision-free motions.

Keywords — Multi-Agent System, Decentralized Collision Avoidance, Minimum Enclosing Rectangle, Rectangular Roundabout, Rectabout, MER Rectabout.

Acknowledgments

I have the pleasure to acknowledge here some of the many people who have inspired, supported, and educated me over the past three years. First and foremost, I am very grateful to my primary supervisor Prof. Ajit Narayanan who gave me the opportunity to start my PhD in Robotics research. In particular, Ajit gave full support, encouragement, guidance from the initial to the final stage which enabled me to develop a deep understanding of the subject. He has a remarkably good taste in research and an excellent sense of strategy. He showed me how to approach a research problem in different ways and find the best solution. I greatly appreciate his open personality, patience, enthusiasm, and immense knowledge that, taken together, make him a great supervisor.

I am very indebted to Saide Lo, the personal assistant of the Head of School of Computing and Mathematical Sciences (SCMS) for all her kindness and the tremendous support, she has offered me since I started my PhD. Also, many thanks to Gordon Grimsey (now retired) who was the technology resources manager of SCMS, Greg Knowles the current technology resources manager of SCMS, Ramon Lewis the technician of SCMS and Terry Brydon the school manager.

I would also like to thank the past and present members of Auckland University of Technology for their support and straight-talking honesty. I thank Associated Prof. Shaoning Pang who helped me a lot with his own background in bioinformatics and evolutionary computation. Dr. Quan Bai, Dr. Waseem Ahmad and Dr. Anuj Bhowmik deserve special acknowledgment for their thoughtful advice, friendship and a lot of insightful discussions. Because they deserve it and are not thanked nearly enough, I would also like to thank the staff of the SCMS, Dr. Weiqi Yan, Dr. Stefan Schliebs, Dr. Paul Leong, Md. Zulfikar Hossain, Md. Akbar Hossain, Raihana Roslan, Abhimanyu Singh Garhwal. Their helpful influence is clear, and has enriched my educational experience immeasurably.

On a personal level, I would like to express my love and gratitude to my parents in China for their understanding, support and encouragement during my PhD research study. My parents have supported me for almost 10 years since I first came to New Zealand. I owe a great deal to my parents who definitely cannot be thanked enough. Without their support and encouragement, I would not be able to complete my study.

Lastly, I offer my regards and blessings to all of those who have supported me in any way during the completion of the study.

Auckland, New Zealand
August 2014

Fan Liu

Chapter 1

Introduction

This chapter introduces the motivation and background of the research, followed by brief introductions to artificial intelligence, multiple agent systems and nature-inspired observation in the context of collision avoidance.

1.1 Scenarios

The motivation for the research on collision avoidance described in this thesis can be summarized in two scenarios: one at a macro level and the other at the micro or nano level.

> *Scenario one: the high volume of traffic is a problem in cities and towns all over the world.*

Broadly speaking, there are three main reasons for this. One is that cars have become more affordable for the average consumer and they are no longer a luxury item, but something that most families expect to own. A second reason is that traffic lights have become more prevalent in recent years, which cause increased traveling time because of the wait at traffic lights even if there are no cars coming from another

direction. The third reason is that humans in general require enough reaction time and therefore distance from other cars for collision avoidance. This means that road capacity is restricted by the need to ensure that there is sufficient space between cars to allow drivers to stop safely in case the car in front stops or a car comes from a different direction that may lead to collision. A typical solution to the traffic congestion problem is to build more roads. However, increasing awareness of the environmental impact of building more roads is making this option less acceptable in many parts of the world.

From an artificial intelligence perspective, a solution is the idea of robot cars [145] where human provides the destination for the car, and the car is responsible for taking human to the destination. During travel, robot cars can determine speeds and routes. It is possible that, due to faster reactions and some communication with other cars, robot cars can be more densely packed on available roads, thus allowing for an increase in car ownership without needing to build new roads. However, even if the solution is a good idea, how do we guarantee the safety of occupants and passengers in other cars? And what happens if communication between cars is somehow affected?

> *Scenario two: consider a patient with a tumour and a future where nanobots can deliver tumour suppressing molecules to the site of the tumour, thereby removing the need for toxic chemotherapy or radiotherapy. A number of nanoparticles are injected into the cancer patient's body and the particles are programmed to search the body for the tumour, use receptors on their surface to dock onto the tumour when found and release its chemical payload at the tumour site. However, as the nanobots close in on their target, the likelihood increases that they will collide with each other and damage*

their valuable payload or receptors.

There are a number of common elements to both scenarios, despite the difference in scale. The first is the ability of the robot car or nanobot to be given a destination and to find their own way to the destination despite any obstacles in their way. In other words, the car or nanobot has to be 'intelligent' and have the capability of planning and reasoning. The second is the lack of central control. Each car or nanobot is responsible for its own behaviour and for getting to the destination. In other words, each car or nanobot is autonomous and will determine actions for itself rather than being told what to do by a remote controller. The third is lack of communication unless absolutely necessary. Each car or nanobot does not need to know what other cars or nanobots are doing in order to determine what it should do. In other words, each car or nanobot uses its own local view to determine its course of action and is not dependent on the views of others or a global view. In any case, adding communication nano-technology to nanobots and thereby increasing the size of the nanobots could severely compromise the ability of the nanobots to get to their target. To realize these two scenarios, the cars and nanobots should ideally be working in a decentralized environment involving no communication.

All this is well known in the AI, robotics and multi-agent literature. Also, the idea of multiple intelligent, autonomous and decentralized robots, cars and nanobots needing to reach their destination is subsumed under 'swarm intelligence' or 'swarm technologies' [87, 86, 52, 12]. However, there is another aspect implicit in the two scenarios above which is not often recognized in the swarm literature: the need for the robot cars and nanoparticles to avoid each other so as not to damage each other. In other words, the fourth and perhaps critical aspect is collision avoidance. Robot cars

and nanobots are physical agents / machine and therefore require collision avoidance capability to reach the goal position.

If we look at birds flocking, ants foraging, fish swarming and pedestrians on a busy street, we observe very little collision and no obvious communication between these agents. The motivation to this thesis is that: so far, relatively little research has been directed towards this - perhaps most critical aspect of intelligent, autonomous and decentralized systems and architectures; the need for the robots or agents to avoid colliding with each other when moving from their start positions to their goal positions.

One piece of evidence for this relative lack of research in collision avoidance comes from a straightforward series of searches using Google Scholar.

1. There are nearly a million hits for 'robots' and 3.5 million hits for 'agents'.

2. There are almost half a million hits for "intelligent" + "robots" and one and a half million for "intelligence" + "agents".

3. There are just under 300,000 hits for "autonomous" + "intelligent" + "robots" and just under 400,000 hits for "autonomous" + "intelligent" + "agents".

4. There are about 17,000 hits for "decentralized" + "autonomous" + "intelligent" + "robots" and nearly 21,000 hits for "decentralized" + "autonomous" + "intelligent" + "agents".

5. Around 5000 hits for "collision avoidance" + "decentralized" + "autonomous" + "intelligent" + "robots" and approximate 4000 hits for "collision avoidance" + "decentralized" + "autonomous" + "intelligent" + "agents" (the first time that the number of hits for 'agents' drops below the number of hits for 'robots').

That is still a lot of lacking research. To specialize the search further:

6. There are 58 hits for “local view” + “collision avoidance” + “decentralized” + “autonomous” + “intelligent” + “robots” and 55 for “local view” + “collision avoidance” + “decentralized” + “autonomous” + “intelligent” + “agents”. In other words, it is quite possible that the hits up until this point have included approaches involving a global view (the view of all robots or agents) even though ‘decentralized’ was included in the search terms. Accessing or maintaining a global view will require some communication and coordination.

   As mentioned above, it is clear from observations of nature that there are many examples of collision avoidance between autonomous, intelligent, decentralized entities, such as birds, ants and human pedestrians. So it would be natural to think that, of the 50+ hits above, the most obvious way to build collision avoidance into robots and agents would be to model collision avoidance on nature. Lastly, final search:

7. There are two hits for “nature inspired” + “local view” + “collision avoidance” + “decentralized” + “autonomous” + “intelligent” + “robots” and three for “nature inspired” + “local view” + “collision avoidance” + “decentralized” + “autonomous” + “intelligent” + “agents”.

These are, of course, superficial results. However, the main point behind the searches still remains and can be stated as follows: while nature-inspired techniques are well adopted in intelligent, autonomous and decentralized robot and agent systems (typically, for swarm behaviour modelling); there is hardly any attempt to draw inspiration from nature on how to get robots and agents to avoid colliding with each

other. In other words, despite the large number of examples around us in nature of how intelligent autonomous entities avoid collision with each other when moving from one point to another, the robotics and agent-based literature has ignored them. The motivation behind this thesis is: to address what appears to be a gap in robot and agent research; modelling the critical aspect of robots and agents avoiding each other through inspiration from nature. Further evidence for this contention is provided in the literature reviews that follow in the next two chapters.

We will discuss which particular aspect of nature inspiration has been the driver behind the thesis especially in the later chapters. Returning to the series of searches above - it is also clear that there are much uncertainties about what counts as being centralized, autonomous and intelligent. There is also uncertainty as to what counts as an agent and when a robot should be an agent, or vice versa. We also need to say what we mean by "collision avoidance" and "nature-inspired". So before exploring our approach in more detail, we need to explain our understanding of many of the terms above to lay an unambiguous foundation for our reported algorithms and results.

1.2 Background of Artificial Intelligence

The field of artificial intelligence, or AI, attempts not just to understand but also to build intelligent entities. Currently, "AI encompasses a huge variety of subfields, such as learning and perception to such specific tasks as playing chess, proving mathematical theorems, writing poetry, and diagnosing diseases. AI systematizes and automates intellectual tasks and is therefore potentially relevant to any sphere of human intellectual activity. In this sense, it is truly a universal field" [111]. Extending the realm of the social world to include autonomous computer systems is now becoming both

possible and necessary through advances in the field of Artificial Intelligence.

In the past several years, AI techniques have become more and more robust and complex and AI researchers have been attempting to discover the implications of multiple autonomous "agents" interacting in the real world. Russel and Norvig [111] define an agent as "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors". Another definition [47] is "intelligent agents as entities that continuously perceive a dynamic environment, reason about and interpret their perceptions, solve problems, and determine actions". "A computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its designed objectives" is defined by Wooldridge [111], and an intelligent agent is further defined to be "capable of flexible autonomous action in order to meet its design objective. Flexible is interpreted to imply that the agents are reactive to their environment, are able to exhibit goal-directed behaviour, and are able to interact with other agents" [111].

Finally, "Distributed Artificial Intelligence (DAI) has existed as a subfield of AI for less than two decades. DAI systems can be defined as cooperative systems where a set of agents act together to solve a given problem. These agents are often heterogeneous (e.g. in a Decision Support System, the interaction takes place between a human and an artificial problem solver). Traditionally, DAI is broken into two sub-disciplines: distributed problem solving and multiple agent systems. [15]"

As can be seen from the above quotes, the concept of agents is closely tied to problem solving, autonomy, co-operation and multiplicity. We now explore these concepts in more details.

1.3 Multiple Agent (Multi-agent) System

Multi-agent systems is the subfield of AI that aims to provide both principles for the construction of complex systems involving multiple agents and mechanisms for the coordination of independent agents' behaviors. Multi-agent systems allow the sub-problems of constraint satisfaction to be sub-contracted to different problem solving agents with their own interests and goals. Furthermore, domains with multiple agents of any type, including autonomous vehicles and even human agents have attracted interest in the field of study [127]. For the purposes of this research, we consider an agent to be an entity, such as a robot, with goals, actions, and domain knowledge, situated in an environment. However, it is believed that much of the prior research in non-robotic multi-agent systems is relevant to robotic multi-agent systems (referred to as multi-robot systems).

The idea of a multiple agent system is inspired by daily human activities, where multiple agents might be independently doing tasks at the same time. The definition of multiple agent system, the relationship between coupled and decoupled approach, and the relationship between multiple agent system and traditional multiple agent collision avoidance algorithms are now discussed.

1.3.1 Definition of Multiple Agent System

Broadly speaking, multiple agent systems are everywhere in nature. For example, ants by themselves may seem to act randomly and without any discernible purpose, but when the collective interactions among ants are taken together, a collective intelligence and behaviour that have the capability of solving many problems, such as finding food and building a nest will emerge. Multiple agent systems can be used to

achieve tasks beyond the capability of an individual agent, especially in the presence of uncertainties, incomplete information, distributed control, and asynchronous computation [56]. The goal of multiple agent systems is to model such natured agents functionality to explore the collective efficiency and effectiveness in relation to specific activities.

A multiple agent system is in principle about lower level agents independently and separately operating on local information to achieve global goals. While the task is handled by independent agents, it may require communication or group cooperation to improve efficiency. Collision avoidance is an important issue in multi-agent systems that involve planning, searching or coordination. According to [73], the definition of decentralized multiple agent system is "agents plan their routes and make decisions independently. Ideally, an optimal decentralized multiple agent system should allow each of the agents involved in a possible collision to make the minimum changes to their planned routes so that - after avoiding collision - they can return to their optimal planned routes.". It is also described as decentralized collision avoidance in the literature [53, 63, 122].

Multi-agent systems differ from single-agent systems in that several agents exist with their own goals or a shared goal. From an individual agent's perspective, multi-agent systems differ from single-agent systems most significantly in that the environment's dynamics can be affected by other agents. In addition to the uncertainty that may be inherent in the domain, the behaviour of other agents can affect the environment in unpredictable ways. Thus, all multi-agent systems can be viewed as being in a dynamic environment. Figure 1.1 shows the schema of multi-agent

systems. Each agent is both part of the environment and modeled as a separate entity. There may be any number of agents, with limited local view (front view), with different degrees of heterogeneity and without the ability to communicate directly.

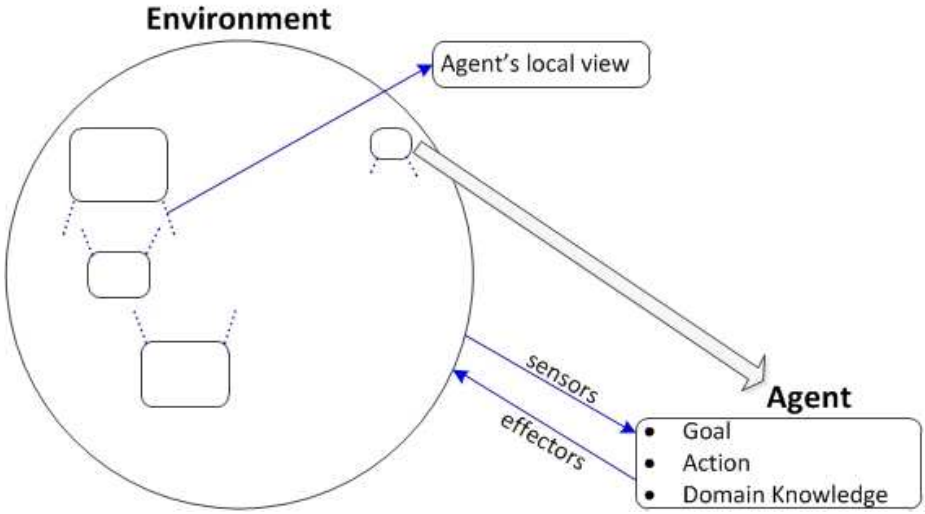


Figure 1.1: Schematic diagram to show the idea for multiple agent systems with local view, no central coordinator, no communication and heterogeneity. Agent models its goal, action, and domain knowledge.

1.3.2 Coupled Approach vs. Decoupled Approach

According to [99], in multi-agent systems, a coupled approach refers to a fixed order motion planning method that can achieve a single search using all agents to find the optimal solution for every agent by treating the agents as centrally controlled entities that follow a strict ordering of movement.

In contrast to coupled approach, a decoupled approach allows agents to plan independently and move in any order. The aim of this research is to explore and evaluate decoupled multi-agent systems with no central control. More detail on a decoupled approach will be presented in Section 2.4.

Note that the definition of multi-agent system also mentions collision avoidance between agents, but this has been often ignored in previous multi-agent approaches [28, 27, 26]. Collision avoidance for multi-agent systems is an important topic [98, 99, 100]. One of the major problems with a multi-agent system where agents perform tasks independently of each other is the need to ensure that agents, after planning their moves, do not collide with each other when converting their planned moves into actual motion. Collision avoidance is necessary due to each agent planning its moves independently of any other agent, thus the risk of collision is not detected until it happens or almost happens. The aim is to ensure that any action taken to avoid collision with another agent in real-time does not adversely affect the 'optimal' planned route beyond the minimum extra effort required to avoid the collision.

1.3.3 Traditional Collision Avoidance Algorithms

Historically, collision avoidance approaches have been focused on robot applications. Previous collision avoidance approaches are based on speed adaptation, route deviation by a single agent only, route deviation by two agents, or a combined speed and route adjustment. Priority-based approaches are widely used in the literature, e.g. [14, 137, 118], but these centralized communicative approaches can become impractical as the number of agents increases. In particular, time complexity can be exponential in the dimension of the composite configuration space.

Many decentralized approaches [63, 107, 123] have been presented recently. However, such techniques require a global view or communication between agents.

Interestingly, there is also a lack of understanding of what "collision" actually

means. As will be shown later in the thesis: it is important to have a clear understanding of what collision types exist if algorithms for collision avoidance are to be effective.

1.4 Nature-Inspired Observation

The use of nature as a source of inspiration is a well established concept in computer science. The past few decades had seen computational approaches that had brought out new ways of thinking during observations in nature. Natural computing [147], also called natural computation, is a terminology introduced to encompass three classes of methods: 1) those that take inspiration from nature for the development of novel problem-solving techniques; 2) those that are based on the use of computers to simulate natural phenomena; and 3) those that employ natural materials (e.g. molecules) to compute. The main fields of research that comprise these three branches are artificial neural networks, evolutionary algorithms, swarm intelligence, artificial immune systems, fractal geometry, artificial life, DNA computing, and quantum computing.

Multi-agent systems consist typically of a population of homogeneous or heterogeneous agents interacting locally with one another and with their environment. The inspiration often comes from nature, especially biological systems. Consider, for example, a group of people walking through a crowded city center or a school of fishes that aggregate in the presence of a predator. In both cases the behavior of an individual is affected by the behaviour of its neighbours.

Bio-inspired robots are robots that resemble living organisms in some specific characteristics, such as the control system, the morphology, the actuators, or the electronics. Robot engineers take inspiration from biology in order to design robots

that have better agility and flexibility, display a novel functionality, more adaptive and intelligent, or that can better operate in the vicinity of humans. For example, snake robots [84] are developed for search and rescue in collapsed buildings where wheeled and legged robots as well as humans may not meet the necessary requirements to move over debris, climb high obstacles, and pass through narrow openings, see Figure 1.2.

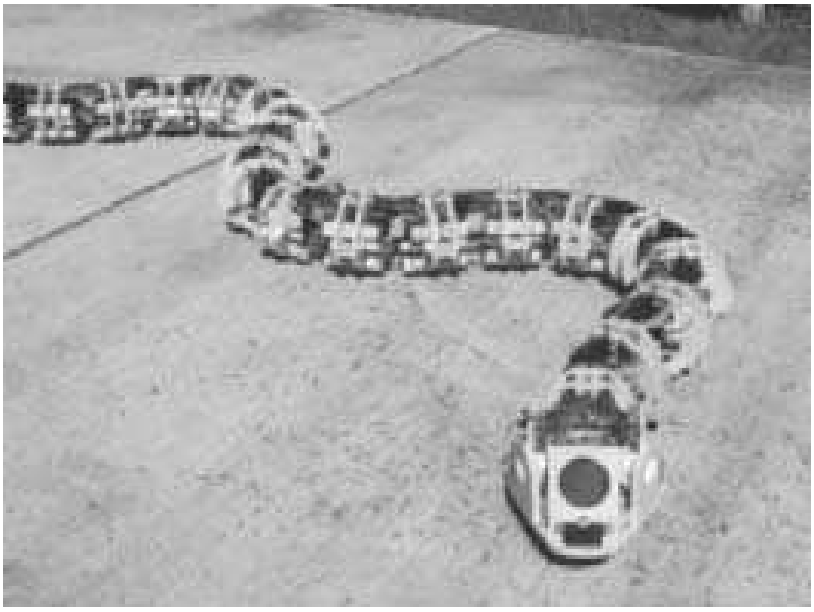


Figure 1.2: A snake robot with compass, sonar, and heat sensors; it may provide better agility and flexibility than wheeled or legged robots for search-and-rescue missions in collapsed buildings (from [84]).

1.5 Motivation for the Presented Research

Having explained our understanding of the main terms that constitute this thesis, we can now rephrase our motivation for the research that follows.

Multi-agent systems is the subfield of Artificial Intelligence (AI) that aims to provide both principles for construction of complex systems involving multiple agents

and mechanisms for coordination of independent agents behaviors. For the purposes of this research, to be an entity an agent is considered as a robot - with goals, actions, and domain knowledge, situated in an environment. The background of the research is the need to develop new methods not just for a small collection of autonomous agents to reach their goals but also for a large number of agents. Centralized control and command work for a small number of agents, but as the number of agents grows, the communication needed between the centralized control and the agents rapidly increases, leading to the problem of agents moving only at the speed at which it takes to communicate with large numbers of agents. Additionally, as the number of agents grows in a constrained space ('configuration space' if the agents are robots), there is an increased risk of collision between agents. The needs to predict and avoid collisions centrally adds a significant overhead to the communication requirements. As noted earlier, if we look at humans going on foot in crowded areas or driving by car in urban traffic, we observe very little collision and no obvious communication between these agents. As a result, new methods need to be found to supplement existing and well-established AI, agent-based and robot movement algorithms for large-scale planning and coordination tasks. The research study proposed here will investigate - how classical AI search algorithms can be adapted to deal with large-scale autonomous agent planning and coordination using the latest knowledge derived from nature-inspired techniques.

The aim of this research is to implement human-like collision avoidance to large scale autonomous agent planning and coordination with no priority, no communication and no global view. The goal here is to evaluate how existing computer science algorithms for finding paths and routes can be adapted to autonomous multi-agent

systems and how agents on a small scale and a large scale can equally adapt suitable motion to find routes without the need for centralized control and command.

1.6 Structure of the Thesis

The thesis is structured as follows:

Chapter 2 contains a review of previous studies on agent research for collision avoidance. In the review, the traditional collision avoidance method which includes single agent and multiple agent path planning model are investigated. In the context of decentralized multi-agent systems, various collision avoidance approaches are summarized, such as prioritized planning and path coordination. The importance of collision avoidance in agent simulation is then analyzed. The limitations of the existing collision avoidance approaches are identified.

Chapter 3 presents an overview of a range of robotic studies on collision avoidance by providing background information. A brief description of existing approaches and modeling that have been used for multi-robot systems is included.

Chapter 4 describes the research methodology along with open questions that are relevant to this research and the proposed collision avoidance method for multi-agent systems, where the the limitations of previous methods and ideal collision avoidance methods are discussed for both collision avoidance and multi-agent systems. This chapter also introduces the idea of the Minimum Enclosing Rectangle (MER) based collision avoidance method, where MER is used as a form of roundabout carrier, allowing agents to replan the path.

Chapter 5 proposes a method to allow agents to replan their routes in real time, taking into account a dynamic environment. This chapter defines various collision types with several possible collision scenarios in a multi-agent system. The remainder of the thesis will deal with collision avoidance based on these collision types.

Chapter 6 proposes novel real time collision avoidance algorithms to solve deadlock situations in tunnel-like environments. This chapter attempts to develop a new algorithm to solve the deadlock situation based on defined collision types.

Chapter 7 presents a new algorithm on formation based on defined collision types in the previous two chapters.

Chapter 8 proposes a novel and dynamic rectangular roundabout ('rectabout') collision avoidance method based on human behaviour for multiple, homogeneous, autonomous and mobile agents. This chapter describes the principle of MER and the idea of MER rectabout. The experiments are conducted in 2 aspects – local behaviour and large scale simulation. Additionally, 3 case studies are demonstrated in aspects of: (1) the capability of collision avoidance; (2) the adaptability of collision avoidance; (3) the scalability of collision avoidance.

Chapter 9 proposes a novel hybrid rectabout algorithm for a group of heterogeneous agents in collision avoidance problems. This chapter briefs the idea of the hybrid rectabout and demonstrates the performance of the proposed collision avoidance method. The experiment is conducted in a real robot demonstration and simulation including heterogeneity and scalability tests. Furthermore, an experimental comparison between the proposed and traditional decentralized

multi-agent collision avoidance algorithm is undertaken.

Chapter 10 concludes the thesis and provides an overview for future work.

1.7 Publications

The following research papers have been written and published during the course of this research thesis.

1. Fan Liu and Ajit Narayanan. (2011) *Real Time Replanning based on A* for Collision Avoidance in Multi-Robot Systems.* In Proceedings of the 8^{th} International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2011), pp. 473-479, 23-26 November 2011, Incheon, Korea.

2. Fan Liu, Ajit Narayanan and Quan Bai. (2012) *Effective Methods for Generating Collision Free Paths for Multiple Robots based on Collision Type (Demonstration).* In Proceedings of the 11^{st} International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012), pp. 1459-1460, 4-8 June 2012, Valencia, Spain.

3. Fan Liu and Ajit Narayanan. (2013) *Roundabout Collision Avoidance for Multiple Robots based on Minimum Enclosing Rectangle (Demonstration).* In Proceedings of the 12^{th} International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), pp. 1375-1376, 6-10 May 2013, Saint Paul, MN, USA.

4. Fan Liu and Ajit Narayanan. (2013) *A Human-Inspired Collision Avoidance Method for Multi-robot and Mobile Autonomous Robots.* In Proceedings of the

16^{th} International Conference on Principles and Practice of Multi-Agent Systems (PRIMA 2013), pp. 181-196, 1-6 December 2013, Dunedin, New Zealand.

Chapter 2

Review of AI Research for Collision Avoidance between Agents

Traditionally, agents do not consider collisions, because collision avoidance is not usually an issue for simulated (or soft) agents. This chapter investigates the importance of collision avoidance for virtual (or soft) agents in simulated environments, as identified in the AI literature. Section 1 presents the outline of virtual agents application. In section 2, path planning in agent simulations is reviewed. Agent path planning is categorized into two groups, with coupled planning and decoupled planning described in detail in section 3 and section 4, respectively. Section 5 covers the current emergent research issues for collision avoidance in multi-agent system. Finally, section 6 summarizes the agent review chapter.

2.1 Introduction

Multi-agent simulation has been widely exploited in a number of virtual environment applications such as games, virtual world, entertainment, and especially in the

computer graphics community. It is also called crowd simulation [142] when the following research issues are addressed: animation, motion planning, rendering, etc. Crowds mean hundreds or thousands of humans or pedestrians, with rendering in real-time [149]. These kinds of applications can provide one with a feeling of being immersed in the dynamic scene, therefore enhancing the reality of the systems. Realistic and believable simulation is one of the important aspects of computer graphics application. Normally, agent simulation is separated into two types: real-time simulation and non real-time simulation [64]. Real-time simulation application in games is often seen, which allows user interaction, while non real-time simulation is normally used in film productions. In non real-time simulation, there is no interaction with users and the simulation occurs without any control of the user. The research on interacting with multi-agent systems in real time is still rare. However, multi-agent systems have great potential in simulated training and safety evaluation by using immersive virtual environments, which includes evacuation in emergency situations [149], military training [142], and urban planning [64].

2.2 Agent Path Planning

Many efficient path planning algorithms have been proposed for both single-agent and multi-agent systems. For example, A* algorithm [89] based on heuristic determination is proposed for single agent path planning. It can plan an optimal path from an initial position to a desired goal position in a static environment with fixed obstacles. Approaches based on a potential field [39, 105] (see Section 3.2.2) are suitable for systems with a static environment containing obstacles. Once the goals are changed, it will require significant re-computation to update the potential fields. As for other

planning approaches, both continuum (macroscopic) models (e.g. [133, 19, 71]) and agent-based (microscopic) models (e.g. [149, 38, 3]) are fast enough to handle the path planning for thousands of agents or even more in real time. However, these models are not specially designed for dynamic interaction. In an interactive multi-agent system, each agent's path should be dynamically re-planned to react to a dynamic environment changing (or user's command).

2.2.1 Single Agent Path Planning

The A* algorithm [89], which is a well-known algorithm for the single-agent path finding, can be formalized as a 4-tuple (G, h, s, t), where G is the search graph, h is a heuristic function, s is the start node, and t is the target node. This can be extended to multiple agents as (G, h, A, S, T), where $A = \{a_1...a_n\}$ is the set of agents in the world, $S = \{s_1...s_n\}$ is the set of start nodes for each agent, and $T = \{t_1...t_n\}$ is the set of target nodes for each agent. Finally, it can be extended to a dynamically searching problem by requiring that agents repeatedly update their motion path using the most recent information (the positions of other agents) in every time step. While agents may search on any graph, in this work the configuration space is assumed as a discrete occupancy grid map (refer further details below), which is commonly used in the literature of agents' path planning research.

2.2.2 Multiple Agent Path Planning

In previous multi-agent simulations, path planning is used to determine a route from an initial position to a desired goal position for each agent in multi-agent systems. Planning approaches can be categorized based on the amount of information used

during the planning process [99] - coupled planning and decoupled planning.

We will give the definition, existing techniques, and issues of coupled and decoupled approaches in the following sections.

2.3 Coupled Approach

2.3.1 Definition

Approaches that use global information and plan directly in a state space X (detailed notations in Section 4.3.1) are called coupled centralized approaches. These approaches treat the agent team as a composite agent system, to which classical single-agent path planning algorithms are applied. For example, the A* algorithm [89, 46] can generate complete and optimal solutions to the multi-agent path planning problem under a centralized and coupled approach. Motion planning algorithms for single mobile agent systems have been intensively studied for years [65, 114, 51]. Examples of classical single agent path planning algorithms include sampling-based planning, potential-field techniques and combinatorial methods. Sampling-based planners [59] avoid the explicit construction of C_{obs} (obstacle space) by sampling different configurations to generate curves that represent collision-free paths in C_{free} (free space); potential field techniques [10, 11] construct real-valued functions that pull the agent toward the goal, and repulse the agent away from obstacles, via a combination of force vector fields. Combinatorial method constructs roadmaps through the configuration space using techniques such as cell decomposition (e.g. [117]).

2.3.2 Problem Modeling

Extending the problem further to multiple agent path planning requires even more computational resource. An example of a centralized approach for generating complete multi-agent path solutions is the work of Parsons and Canny [101], which takes a global cell decomposition approach, incorporating obstacles and other agents in a unified configuration space representation. This algorithm first computes a decomposition of the free space into cells; it then searches through the resulting adjacency graph for a path. However, the algorithm is exponential in the number of agents. Other centralized algorithms that represent the path planning problem as a cross product of the configuration spaces of the individual agents also exist [11, 113].

Because of the high dimension of the multi-agent configuration space, centralized approaches that treat the multi-agent team as a single composite agent tend to be computationally impractical if the full search space is used. Instead, techniques that reduce the size of the search space have been shown to be practical for small-sized problems. One way to reduce the search space is to constrain the allowable paths that agents can follow by limiting the motion of the agents lying on roadmaps in the environment. Intuitively, roadmaps are akin to automotive highways, where agents move from their starting position to a roadmap, move along the roadmap to the proximity of the goal, and then move off the roadmap to the specific goal location. More formally, a roadmap is defined as follows [20]:

Definition (Roadmap): A union of one-dimensional curves is a roadmap (RM) if for q_{start} (the configuration of all start positions) and q_{goal} (the configuration of all goal positions) in C_{free} that can be connected by a path, the following properties hold:

1. **Accessibility:** there exists a path from $q_{start} \in C_{free}$ to some $q'_{start} \in RM$,

2. **Departability:** there exists a path from $q'_{goal} \in RM$ to $q_{goal} \in C_{free}$, and

3. **Connectivity:** there exists a path in RM between q'_{start} and q'_{goal}.

Typically, a roadmap RM is represented as a graph $G = (V, E)$, in which the nodes V represent collision-free configurations and the edges E represent feasible paths. A feasible path is one that can be executed by agent A^i, based on its motion constraints. Various algorithms have been created that make use of the roadmap concept for motion planning, both for single agents and for multi-agent teams (e.g., [112, 103, 129]).

2.3.3 Existing Techniques

It is worth noting that many other roadmapping approaches to multi-agent path planning have been proposed. For example, the work of Ryan [112] reduces the search space by decomposing the original map into subgraphs, planning paths between subgraphs, and then coordinating motions within the subgraphs. This approach has been shown to be effective for up to 10 agents. In [22], Clark et al. introduce the concept of dynamic networks, which are formed between agents that are within communication range. Within this framework, only agents within the same network use a centralized planner, which is based upon probabilistic roadmaps [59]; otherwise, agents plan their paths using decoupled planners based on optimizing priorities. Some of the existing coupled approaches are explained as follows:

Super-graph Method

In [129], Svestka and Overmars present an approach for creating a composite roadmap, which represents a network of feasible motions for the composite agent. This composite roadmap is created as follows: (a) a roadmap for each individual agent is constructed using the standard roadmap generation algorithm, Probabilistic Path Planner (PPP) [59]; (b) n such roadmaps are combined into a roadmap for the composite agent, which can be used to generate coordinated paths.

Specifically, the coordinated path for the composite agent $(A^1, \ldots, A^n)$ is an n-tuple of paths feasible for all agents A^i that, when executed simultaneously, introduce no mutual collisions between the individual agents. Formally, let $C^{[0,1]}$ represent the configuration space from time $t = 0$ to time $t = 1$, where the agent is at its starting position at time 0, and is at its goal location at time 1. Let $s_1, \ldots, s_n$ and $g_1, \ldots, g_n$ be given starting and goal configurations for the n agents, where $\forall i \in \{1, \ldots, n\}$: $s_i \in C_{free} \wedge g_i \in C_{free}$. Let P represents a free path if P is in C_{free} for all times t (i.e., $\forall t \in [0, 1] : P(t) \in C_{free}$). Let $A \cap B \neq 0$ (i.e., A and B intersect) be represented by $A \otimes B$. Then if $P_1, \ldots, P_n \in C^{[0,1]}$ are feasible paths, such that for all $i, j \in \{1, \ldots, n\}$

1. $P_i(0) = s_i \wedge P_i(1) = g_i$

2. $i \neq j \Rightarrow \forall t \in [0, 1] : \neg A(P_i(t)) \otimes A(P_j(t))$

then $(P_1, \ldots, P_n)$ is a coordinated path for $(A^1, \ldots, A^n)$ solving the problem $((s_1, \ldots, s_n), (g_1, \ldots, g_n))$.

Svestka and Overmars present an approach for constructing such a coordinated path for a composite agent [129]. The basic idea is to seek paths along the roadmap, G, that allow the agents to move from their starting to their goal configurations, while

disallowing simultaneous motions or motions along paths that are blocked by other agents. This type of path is called a G-discretized coordinated path. The authors introduce the concept of super-graphs, which represent roadmaps for the composite agents created by combining n simple agent roadmaps. Two variants of super-graphs are proposed – flat super-graphs and multi-level super-graphs.

In the flat supergraph, a node represents a feasible placement of the n simple agents at the nodes of G, and an edge represents a motion of exactly one simple agent along a non-blocked path of G.

The second type of super-graph – the multi-level super-graph – reduces the size of the super-graph data structure by combining multiple nodes into a single node of the graph. This approach makes use of the concept of subgraphs. Whereas the nodes in a flat super-graph represent agents being located at particular nodes of G, the nodes in a multi-level super-graph represent agents being located in a subgraph of G. The restriction placed on node combinations is that the resultant subgraphs should not interfere with each other, meaning that the nodes in one subgraph cannot block paths in another subgraph. Experimental results have shown that the multi-level super-graphs are typically much smaller than the equivalent flat super-graphs.

Svestka and Overmars apply this approach to teams of up to 5 car-like agents in simulation. An example of these results is shown in Figure 2.1, illustrating the feasibility of this approach for small-sized multi-agent teams. Nevertheless, this type of approach is appropriate only for relatively small numbers of agents. For much larger sizes of agent teams, decoupled approaches are necessary, because a disadvantage of the supergraph is that its size is exponential according to the number of agents.

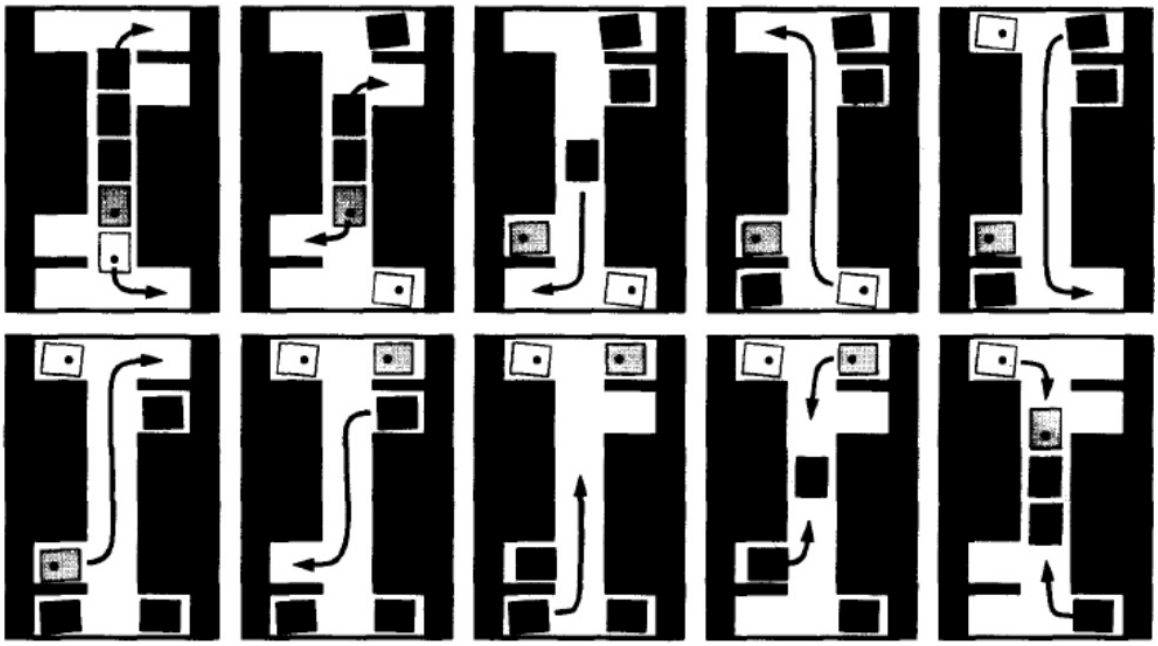
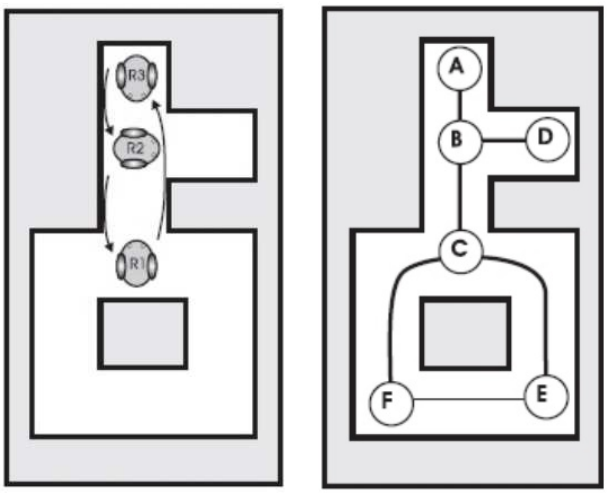


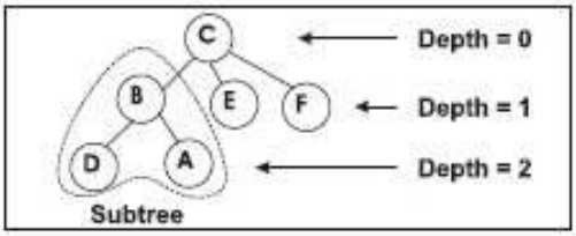
Figure 2.1: An illustration of a coordinated path generated by the super-graph approach, for 5 nonholonomic car-like agents (from [129]).

Spanning Tree Method

Peasgood et al., [103] present another roadmap-based planner for multi-agent teams. This approach is a multi-phase planner that uses a graph and spanning tree representation to create and maintain obstacle-free paths through the environment. Initially, a graph is created, in which the nodes are the agents' initial and goal positions, and the edges represent the connectivity of the node positions. An example is illustrated in Figure 2.2(a), in which the starting positions of the three agents (R1, R2, and R3) are (C, B, A), while the goal positions are (A, C, B). Figure 2.2(b) shows the graph-based map for this example. Then, a spanning tree of this graph is created, which is a connected subset of the original graph that includes all the nodes without cycles; Figure 2.2(c) shows an example of a spanning tree. The root of this spanning tree is chosen to be the node that is closest to the geographic center of the map.



(a) Original planning problem. (b) Graph-based map.

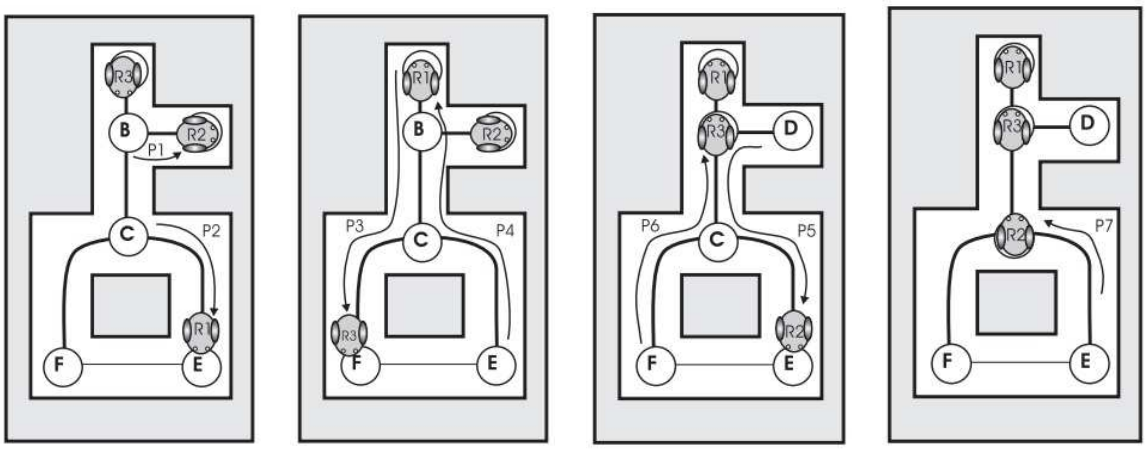


(c) Spanning tree for the graph representation.

Figure 2.2: An example of a multi-agent path planning problem using the spanning tree method of Peasgood, et al., along with the corresponding graph and spanning tree (from [103]).

In the first phase of the approach, a plan is generated that moves the agents to the leaves of the spanning tree along collision-free paths, as shown in Figure 2.3(a). In the second phase, the agents are moved into positions where they can reach their goals without creating obstructions for other agents. This is accomplished by processing the agents in order according to the depth of their goals in the spanning tree. This is shown in Figure 2.3(b and c). The third phase moves agents to the remaining unfilled goal locations, as shown in Figure 2.3(d). These three phases result in a

sequence of motions that allow only one agent to move at a time. The final phase of the process seeks to improve the quality of the concurrent plan by allowing agents to move simultaneously when doing so does not introduce any collisions.



(a) (b) (c) (d)

Figure 2.3: The multi-phase solution of the multi-agent path planning problem, using the spanning tree method of Peasgood, et al. (from [103])

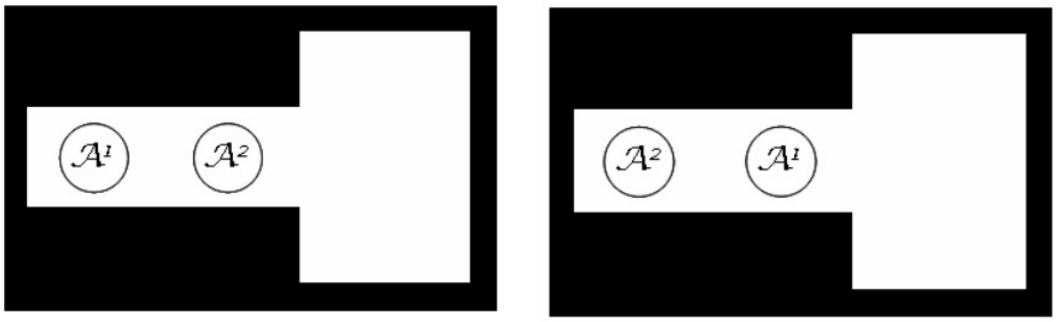
Peasgood et al., [103] show that this algorithmic approach results in time complexity that is linear in the number of agents. To further improve the resulting path lengths, the authors propose a hybrid planning approach, which uses the regular multi-phase planner, but then also uses a decoupled planner (such as [13]) in attempt to find shorter path solutions. For smaller-sized robot teams (less than 20), the decoupled planner can often find better solutions. However, for larger-sized teams, the multi-phase approach is more time-efficient (increasingly as the team size grows larger).

2.4 Decoupled Approach

2.4.1 Definition

According to [99], decoupled approaches decompose the path planning problem into independent components that can find good solutions quickly, although at the cost of losing optimality (i.e. the quality of the resulting solution) and completeness (i.e. whether they are guaranteed to find a solution if one exists). Decoupled approaches can either be centralized or decentralized: centralized decoupled (or semi-centralized) approach is a coordinator-based technique where all agents plan the path independently, and then report to the coordinator regarding their intentions. Coordinator then gives a command to accept or reject their intentions for collision avoidance, because the agents here might only have local views. Decentralized decoupled approach is a distributed multi-agent system where every agent attempts to solve the problem locally with / without communication. Most commonly, approaches plan individual paths for agents, followed by methods for handling collision avoidance. For instance, Figure 2.4 shows an example of a situation that is difficult for decoupled approaches to solve. In this situation, agents must exchange positions in a narrow corridor. A centralized approach on the other hand would find a solution in which the agents first move into the open space at the end of the corridor to exchange places.

Path planning builds a bridge to exploit the relationship between traditional decoupled planning algorithms and various realistic dynamic environment settings, in order to avoid collisions between agents via planning, evaluating, and resolving [59, 58]. Based on the characteristics of various path planning bridges, the different approaches to decoupled planning in the literature can be divided into two categories [99, 65, 66]



(a) (b)

Figure 2.4: An example of a multi-agent path planning problem that is difficult for decoupled approaches to solve (from [65]).

– prioritized planning and path coordination. Prioritized planning considers the motions of the agents one at a time, in priority order, calculating path information for the i^{th} agent by treating the previous i - 1 agents as moving obstacles. Path coordination, on the other hand, first plans independent paths for the agents separately, then seeks to plan their velocities so as to avoid collisions along those paths.

2.4.2 Prioritized Planning

The prioritized planning approach to multi-agent path planning was first proposed by Erdmann and Lozano-Pérez [32]. In this approach, priorities are assigned to each agent. These priorities could be assigned randomly, or they could be determined from motion constraints, in which more-constrained agents are given higher priority. A path is planned for the first agent using any single-agent path planning approach. The path for each successive agent, A^i, then takes into account the plans for the previous agents $A^1, \ldots A_{i-1}$, treating these higher-priority agents as moving obstacles.

More specifically, in the prioritized planning approach of [32], the configuration

space is extended to account for time, since the time-varying motions of previously planned agents must be taken into account. Configuration space-time is represented as a list of configuration space slices at particular times, specifically, those times corresponding to when a moving object changes its velocity. Motions between slices can then be interpolated via straight-line translations between these configuration space slices. The configuration space-time can be constructed in $O(m)$ time, where $m = nr$, for n edges in the environment and r time slices.

In another example, a prioritized A* algorithm [14] is extended to multi-agent systems based on a priority scheme. An optimal route is first calculated for each agent using A* [89] (the decoupled phase) and then agent paths are checked for possible collisions (the centralized phase) with a global priority scheme for detecting and resolving collisions. However, this type of planning approach requires computation time that is exponential in the dimension of the multi-agent configuration space. Thus, these approaches can only be used in real time for the smallest of problems (scalability). Another limitation with these approaches is that the global environment is still considered as a static configuration. It is not clear how effective this coupled and centralized approach can handle dynamic environments with priority, such as new obstacles and other agents appearing.

Other researchers who have studied prioritized path planning for multiple mobile agents include [34, 143, 14, 18]. Both Ferrari et al. [34] and Warren [143] use a fixed priority scheme for the decoupled planner. In the work of Buckley [18], a heuristic determination is applied to assign higher priorities to agents that can move in a straight line to their target location. Chun et al. [21] use this priority scheme to coordinate independently-generated schedules online, as the conflicts arise. The work

of Azarm and Schmidt [7] considers all possible priority assignments, although the resulting approach is computationally complex.

Priority-based approaches are widely used in the literature, e.g., [14, 118, 137]; the advantage of prioritized planning approaches is that they reduce the problem from a single planning problem in a very high-dimensional space to a sequence of planning problems in a much lower dimensional space. The disadvantage, as with all decoupled approaches, is that these approaches are not complete, and centralized approaches can become impractical as the number of agents increases. In particular, time complexity can be exponential in the dimension of the composite configuration space.

2.4.3 Path Coordination

In the path coordination approach, the path planning step first generates individual agent paths independently, using common single-agent path planners. The second step plans a velocity profile that each agent should follow along its path so as to avoid collisions with other agents. This approach is typically called fixed-path coordination [99], since the paths planned in the first step are not altered in the second step. Instead, only the velocities taken by the agents along the paths are varied.

A formal explanation assumes that the path generated for each individual agent in the first step constrains agent A^i to follow a path $\tau_i : [0,1] \rightarrow C^i_{free}$. Then, an m-dimensional coordination diagram $X = [0,1]^m$ for m robots is defined that is used to schedule the motions along their paths so that they do not collide [90]. The i^{th} coordinate represents the domain, $S_i = [0,1]$, of the path of agent A^i. At state $(0,\ldots,0) \in X$, every agent is in its initial starting configuration. At state

$(1, \ldots, 1) \in X$, every agent is at its goal configuration. Within the coordination diagram, obstacles form obstacle regions X_{obs} that must be avoided. Any continuous, obstacle-free path, $h : [0, 1] \rightarrow X$, for which $h(0)=(0, \ldots, 0)$ and $h(1)=(1, \ldots, 1)$, is a valid path that moves the agents from their starting positions to their goals. The objective, therefore, is to find $h : [0, 1] \rightarrow X_{free}$, in which $X_{free} = X \setminus X_{obs}$, and $\setminus$ means removing or excluding.

Many other researchers have looked at variations of the path coordination approach. In [68], Lee and Lee use a similar idea to plan the motions of two agents. Griswold and Eem [41] take uncertainty of the moving obstacles into account while using the same principle for path planning. Pan and Luo [95] use the concept of traversability vectors to analyze the spatial relationship between the agent and moving obstacles, and develop a search algorithm to coordinate the agent motion. Rude [110] proposes a space-time representation for collision avoidance in pre-planned individual agent paths. In [44], Guo and Parker present a decentralized path coordination approach that also incorporates optimization issues into the planning, including a global performance measurement to minimize the weighted sum of the most time taken to reach the goals and all idle time, as well as individual optimization goals for navigation over rough terrain. In [67], LaValle and Hutchinson consider multiple agents with independent goals and performance measures, and propose algorithms optimizing a scaling function that is a weighted average of individual performance functions. Direction maps [54] add penalties to the heuristic function by storing a direction for each node. An agent gets a penalty for not moving in the exact direction that is stored in the node. The penalty is proportional to the difference between the direction stored in the node and the actual direction the agent is moving in. The

direction for a node changes when an agent moves across it, and the new direction is a weighted value between the old direction and the direction the agent is moving in. Along time, the direction map will form lanes to discourage agents from moving against the general flow of other agents.

While all of these decoupled approaches typically allow good solutions to the multi-agent path planning problem, they can lead to deadlocks, in which solutions cannot be found. In these cases, it may be possible to make use of a centralized planner for small portions of the original problem in order to solve the immediate deadlock problem.

2.5 Research Issues

Many open issues in multi-agent path planning and coordination remain. The following is a discussion of open research issues [99, 90, 53, 76, 63].

- Existing multi-agent path planning approaches ignore sideswipe collisions among agents (i.e., they only consider the collision in which two agents try to occupy the same node during the same time-step) [54, 120, 126], and allow diagonal movement between two adjacent nodes. However, in many real world applications, sideswipe collisions may also block agents' movements or cause deadlocks. Therefore, it is absolutely crucial to investigate all possible collision scenarios in a multi-agent system and design space efficient collision avoidance techniques when agents are moving. (Chapter 5)

- Deadlock is a popular issue in multi-agent path planning [90, 108, 53]. If two agents move towards each other in one narrow passageway, the lack of space

in the tunnel will prevent one agent from moving out of the way. Ideally, agents should temporarily cooperate, leading to one agent re-planning a way to escape this deadlock while the other agent can continue to move to its target location. (Chapter 6)

- Formation and flocking have been topics of interest in multiple mobile agent systems since the inception of the field. A key question in both flocking and formation control research is determining the design of local control laws for each agent that generates the desired emergent collective behavior, and how paths can be planned for a moving goal in multi-agent formations [87, 86]. (Chapter 7)

- Most previous approaches are based on the classical path planning model, such as global information data structures, priority schemes and communicative methods. This research aims to develop and implement: a human-like autonomous collision avoidance approach for the multi-agent system using non-priority, local views and a configuration space where an agent can make any move it likes as long as there is no obstacle (fixed or dynamic) in the way. In such a configuration space there are no predetermined routes. Every space in the configuration space is reachable from every neighbouring space provided there is no obstacle. The scalability issue is also taken into account. How can a large number of agents autonomously and equally deconflict paths efficiently in dynamic and uncertain environments? (Chapter 8)

- There is also an issue of collision avoidance for heterogeneous agents [123, 135, 124]. Collision avoidance requires adaptive local views since the variable speed of agents means not just that agents are moving at different speeds from each

other but also that an agent can vary its own speed while moving. (Chapter 9)

The above issues represent great challenges to collision avoidance in multi-agent path planning systems. There is a real need to address the above research problems. However, as can be seen from the literature review above, there has been no attempt so far to model collision avoidance on models adapted from collision avoidance in nature. This 'blind spot' mentioned in Chapter 1 has been shown to exist through a systematic review of the agent-based literature and, in the next chapter, we shall show that it exists in the robot literature as well.

2.6 Summary

This chapter has reviewed different path planning approaches including single agent and multiple agent path planning, coupled and decoupled approaches. Current techniques typically do not scale well to very large numbers of agents (e.g., thousands), and many still have limitations for extensions to human-like autonomy (e.g., autonomous agents – collision avoidance with local view, no priority and no communication). Developing path planning and motion coordination techniques that incorporate practical motion and sensing constraints of physical robots is still an open issue. Integrating these techniques into physical robots remains uncommon, due to the practical need to integrate these path planning and coordination algorithms with complete sensing, navigation and reasoning systems, as well as the practical difficulty of experiments involving large numbers of robots (e.g., the application of swarm robots) and fallible robots (e.g., robot malfunction).

Chapter 3

Review of AI Research for Collision Avoidance between Robots

Artificial intelligence is the simulation of intelligence in machines [146]. A robot is a mechanical or virtual agent [148]. In the robotics field, especially in the collision avoidance issue, intelligence incorporates / includes autonomy as well as planning, reasoning and cognitive abilities. Achievements in artificial intelligence and robotics include autonomous cars [145]. Robots need agent research because an intelligent agent is a system that perceives its environment and takes actions which maximize its chances of success. Physical robots must consider the collision issue because they are physical and entities are not simulated agents. This chapter reviews the existing approaches of collision avoidance in multi-robot systems. Section 1 introduces the difference between motion coordination (physical robots) and path planning (soft agents). The collision avoidance approaches are reviewed in section 2. Finally, section 3 summarizes the robot review chapter.

3.1 Introduction

Research problems and challenges that have arisen in the area of multi-robot systems have been addressed by using well established path planning and motion coordination approaches. Multi-robot path planning / path coordination is to plan and / or coordinate the complete paths of all of the robots in advance. By contrast, motion coordination focuses on decentralized, online approaches that allow robots to avoid and / or resolve conflict as the situation arises during path execution, such as through the use of traffic control rules. In traffic control applications, individual robots still have independent starting and goal positions, and must move so as to avoid conflict with each other. In this chapter these theoretical foundations are reviewed.

3.2 Existing Approaches

Motion coordination solutions are categorized into traffic control approaches, reactive approaches and swarm techniques. In traffic control solutions, the idea is to pre-define traffic control rules that robots must obey as they move through the workspace. On the other hand, in reactive-style solutions, techniques from potential fields have been adopted to achieve fast, real-time solutions. In swarm techniques, the aim is to control robot motions in order to achieve a group objective, such as maintaining a formation.

3.2.1 Traffic Control Approaches

Typically, traffic control approaches require that individual robots move along paths to their goals that they pre-plan in advance, based only on the individual robot's goals. Then, as regions involving shared resources are reached (such as the space in

an intersection), robots follow the traffic or control rules to coordinate their motions with other robots who also need access to the shared resources [100].

An early traffic control example is proposed by Grossman [42] whose work addresses the motion of large numbers of automatic guided vehicles in a factory. Grossman defines three types of control possibilities:

1. Option 1 - centralized: controls all automatic guided vehicles' paths using centralized traffic control;

2. Option 2 - suboptimal: restricts the roads so that there is a unique route between all starting and goal positions;

3. Option 3 - decentralized: allows automatic guided vehicles to select their own routes autonomously.

Grossman shows that allowing automatic guided vehicles to select their routes autonomously (option 3) is preferred over the sub-optimal restriction of roads (option 2). On the other hand, the centralized approach (option 1) has high combinatorial complexity.

The problem in [42] is formulated as follows. A set of m automatic guided vehicles are allowed to follow unconstrained paths in two dimensions, on a grid-iron network of roadways, with n parallel roads along each axis. Each section of roadway between intersections is called an arc; in this formulation, there are $2n(n-1)$ arcs in the network. Each intersection of roadways is called a node, representing the locations of machine tools to be serviced by the robots. It is assumed that $1 \leq m \leq n^2 - 1$, and that all vehicles move at the same speed, v. Each automatic guided vehicle has the task of moving from a source location (i.e., starting position) to a sink location (i.e.,

a goal location). Defining S to be the average number of time steps per task for each vehicle, the average throughput of all the vehicles together is $W = \frac{vm}{S}$. This throughput must exactly match the throughput of all the n^2 machine tools, leading to a requirement that the vehicle speed must satisfy: $v = \frac{Sn^2}{m}$. The price of m vehicles is considered negligible in comparison to the price of the machine tools. Thus, the problem is formulated as the problem of optimizing the traffic control and the value of m so as to minimize v in an $n \times n$ grid-iron floor plan. The constraints on the traffic in this environment are as follows:

- At the end of each step, one automatic guided vehicle at most may be at each node.
- During each step, no two automatic guided vehicles may pass on the same arc.
- All automatic guided vehicles have equal priority.

There are many variants on the traffic control and conflict resolution theme [57, 6, 151, 139, 72, 78]. For example, Kato et al. [57] categorize traffic rules into three types: first, traffic rules to be applied to the current positions of the robot, such as passage zone, stop, slow. The second is traffic rules to be applied to current positions and conditions, such as overtaking, avoiding obstacles, crossing intersections. Lastly are the traffic rules to ensure safety in case of accidents or failures. These rules are illustrated for robot teams operating in indoor hallway-types of settings. Asama, et al. [6] propose two basic rules for avoiding collisions: if the colliding robot is near the front and approaching, then it should avoid the other agent from the left. Else, if the colliding robot is near the front and leaving, then it must stop for a while. These rules are combined with a communication-based negotiation process that resolves conflicts

by setting priorities based on the task requirements, the environmental situation and robot performances. Another approach to conflict resolution is to use techniques from distributed computing, as shown in [139], in which robots use a mutual exclusion protocol to compete for the right to move along certain pathways or to resolve conflicts at intersections.

In [94], the work considers a more realistic kinematic model of the robot dynamics, recognizing that most robots cannot stop instantly in order to avoid collisions. This model focuses on large numbers of robots (e.g., 70) operating closely in shared, open spaces. This approach is particularly relevant for applications of aerial vehicles flying at a constant altitude. This work makes use of the concept of a reserved region, which is an area over which a robot claims exclusive ownership. The control policy is defined for a set of discrete modes of operation, including a hold state in which a robot is stopped, a straight state in which the robot is moving forward without turning, and two roll states – one for wide turns and another for tight turns.

More recent work in conflict resolution for multi-robot teams is the work of Lalish [63], a global deconfliction maneuver which mimics a 'rules of the road' approach such as roundabout passing behaviour. In this approach all vehicles turn the same way until a conflict-free state is reached for the whole system. The approach is 'global' because the positions of all vehicles are regulated by a deconfliction maintenance controller even if they are not involved in collision, and 'distributed' because collision avoidance computation is shared among a group of robots.

3.2.2 Reactive Approaches

One common reactive method makes use of potential fields [60]. In the potential field approach, the robot moves through space as if it is being acted upon by a set of forces. Attractive forces pull the robot toward a goal destination, while repulsive forces push the robot away from obstacles and / or other robots. At each point in the configuration space, the robot moves along the vector representing the combined forces acting on that point in the configuration space. These concepts have been applied to various multi-robot applications [143] including multi-robot soccer [69]. Other potential field approaches to multirobot coordination include [25, 70, 141, 140].

While all of the above techniques work well for relatively unconstrained situations, they are not analyzed formally to provide guidance for setting the navigation parameters. On the other hand, a more formal method for determining reactive collision avoidance parameters is proposed in [36]. The approach is a collision avoidance method based on an adaptive navigation technique, in which the navigation law is given by a first-order differential equation. Navigation of the robot to the goal and obstacle avoidance are handled by switching the direction angle adaptively. Robots are assigned priorities to determine which vehicles must yield to the others. The proper value of the direction angle is calculated theoretically, based on three robot modes of operation: the navigation mode where the robot is moving towards the goal without interference; the cooperative avoidance mode, in which the robot avoids other robots; and the final mode is when the robot is approaching the goal. The approach has been implemented in simulation for up to five mobile robots and on two physical robots.

More recent works in the reactive technique for multi-robot systems is proposed by

Jur van den Berg et al., [135, 136, 135] involving a velocity-based collision avoidance approach. This lets a robot take just half the responsibility for avoiding a collision, while assuming the other robot reciprocates by taking care of the other half. However, the reciprocal velocity obstacle approach can cause robots to select oscillating velocities as a result of not reaching agreement on which side to pass each other. The latest work (hybrid reciprocal velocity obstacle [124]) reduces the possibility of oscillations and introduces priority to moving left.

3.2.3 Swarm Techniques

Many types of swarm behaviors have been studied, such as foraging, flocking, chaining, search, herding, aggregation and containment. The majority of these swarm behaviors deal with spatially distributed multirobot motions, requiring robots to coordinate motions either (1) relative to other robots, (2) relative to the environment, (3) relative to external agents, (4) relative to robots and the environment, or (5) relative to all (i.e., other robots, external agents and the environment) [98]. Coordinating the motions of robots relative to each other has been a topic of interest in multiple mobile robot systems since the inception of the field, especially regarding flocking and formation control problems. In these problems, robots are assumed to have only minimal sensing, computation, effector and communications capabilities. A key question in both flocking and formation control research is determining the design of local control laws for each robot that generate the desired emergent collective behavior. Other issues include how robots cooperatively localize themselves to achieve formation control [87, 86].

Early solutions to the flocking problem in artificial agents are proposed in [109],

which is a rule-based approach. Similar behavior-based or rule-based approaches have been used in physical robot demonstrations and studies [82, 9]. These earlier solutions are based on human-generated local control rules that are demonstrated to work in practice. More examples include [52, 12, 132, 33, 80, 37, 130, 4].

To conclude this part of the literature review, it has been shown that, while collision avoidance is well-understood and researched in the robot literature, there is very little previous research on adapting nature-inspired collision avoidance techniques to robot collision avoidance. It is not within the scope of this thesis to ask why this blind spot exists, although it could be hypothesized that reasons may include perceived irrelevance of nature inspiration, lack of understanding of how natural entities avoid each other and perceived difficulties in implementing nature-inspired collision avoidance strategies safely and reliably. While it is not within the scope of this thesis to cover the issue of the largely ignored nature-inspired collision avoidance, it is certainly within the thesis's interest to understand how natural entities avoid each other, demonstrate working systems of collision avoidance strategies and to ensure that such systems are safe and reliable for all entities. These points will be covered during the course of the thesis.

It may be noted from the literature review that there is another perceived gap - understanding what a collision actually consists of. This is surprising, given that collision avoidance algorithms need to be shown to handle the specific types of collision they are designed for. The lack of a formal framework for describing collision types also makes it difficult to compare collision avoidance algorithms with each other. The collision types need to be addressed as part of the research methodology if the claims regarding the collision avoidance algorithms actually work and needs support.

3.3 Summary

From a practical perspective, many real-world applications can potentially benefit from the use of multiple mobile robot systems. Examples of applications include container management in ports [2], extra-planetary exploration [128], search and rescue [55], mineral mining [116], transportation [131], industrial and household maintenance [100], construction [121], hazardous waste cleanup [97], security [45], agriculture [106] and warehouse management [48]. To date, relatively few real-world implementations of these multi-robot systems have occurred, primarily due to the complexities of multiple robot systems and the relative newness of the supporting technologies.

From a scientific perspective, understanding interactions between multiple autonomous robots might lead to insights in understanding other types of complex systems, from natural interactions in biology and social systems to engineered complex systems involving multiple interacting agents. Because multi-robot systems operate in stochastic and unpredictable settings, the study of the interaction dynamics in these settings can lead to discoveries that have a broader impact on a wide range of complex non-linear systems [99].

In the next chapter, a description of the research methodology will be given which provides the experimental framework for the research in this thesis.

Chapter 4

Research Methodology

The scientific research methodology (SRM) provides the most logical and constructive way to conduct experiments when there is little previous research on which to base the new research questions and experiments. Section 1 gives a general overview of SRM. The research problems will be formulated and open questions in this thesis study are briefed in section 2. Section 3 gives the proposed method along with the discussion of an ideal collision avoidance method. A design of the experimental method is proposed in section 4. Section 5 provides the analysis of results and validation. A review and evaluation of output reports are given in section 6. Lastly, section 7 forms the summary of this chapter.

4.1 Introduction

Chapter 1 introduced two scenarios to provide a motivation for the research described in this thesis, which is to address a gap in the multi-robot and multi-agent literature: the use of nature-inspired techniques for collision avoidance. Chapters 2 and 3, through literature review, provide evidences of this gap, with much work focused on

rigorous and formal ways to avoid collision based on well-known search strategies and methods for representing the search space within which robots and agents must move. Therefore, the following research question is imposed. Can nature-inspiration be used in multi-robot and multi-agent systems, and if so, how? To answer this question systematically within the scope of a PhD thesis, it requires a research methodology to ensure that the question can be broken down into manageable sub-parts. Also, such a break-down will allow for the emergence of sub-questions to the main research question. Finally, such a breakdown will allow a 'flow' of research that represents a journey of discovery, given that the thesis appears to be located in a space that has not been previously occupied by any PhD thesis or, to the best knowledge, any previous research directly related to the research question. In other words, this thesis does not claim to provide a complete answer to the research question but does claim to be setting down some fundamental pointers as to how the research question, if adopted by others, can be pursued. Along the way there will be a number of hypotheses to test, with results from the tests leading to refinements of hypotheses or new hypotheses to test. This chapter identifies the research methodology adopted to manage and keep control of the research. The concluding chapter will return to the research methodology and will ask whether it has been successful.

The critical aspect of any suitable research methodology, given our research question above and the relative lack of previous work in the area covered by the research question, is the need to formulate a hypothesis, derive some test conditions, implement software solutions relevant to those test conditions, run the software, collect the results, analyze the results against the test conditions, report the results and then either reformulate the hypothesis or derive some new test conditions, and so on. This

process is repeated until, ideally, the best software solution is found for the best test conditions (extremely unlikely in any science) or, as is the case in this thesis, time runs out. When time runs out, the merit of what has been achieved needs to be evaluated both on its own terms (e.g. novelty and significance) as well as the possibility of continuation. We will return to discussing the merit of what has been achieved at the end of the thesis.

4.2 Scientific Research Methodology

The scientific method is a body of techniques for investigating phenomena, acquiring new knowledge or correcting and integrating previous knowledge [40]. The scientific method is a way to ask and answer scientific questions by making observations and doing experiments. Common steps of the scientific methods are:

- To ask questions;
- To do background research;
- To construct a hypothesis;
- To test a hypothesis by doing experiments;
- To analyze data and draw a conclusion;
- To communicate results.

The main challenge in this research is the lack of previous work on autonomous agents for collision avoidance, which involves no priority, communication, global-view and central coordinator. There is no previous work on which to base a fully

autonomous method through nature-inspired observation. The design process consists of a set of steps that starts first with the identification of a problem or a need and leads to creating and developing a solution that solves the problem or meets the need. The steps of the methodology used in this thesis are:

- To define the problem;
- To do background research;
- To design the method;
- Experiments;
- Analysis and validation;
- Output report;
- Review and Evaluation;
- To re-define the questions;

The research method to be adopted in this thesis is illustrated by the diagram in Figure 4.1.

4.3 Research Problems and Open Questions

This section formulates the collision avoidance problems in multi-agent systems and briefs open questions.

Figure 4.1: Reasoning Cycle - Scientific Research.

4.3.1 Research Problems

The multi-agent path planning problem with collision avoidance is defined as follows: given a set of m agents in k-dimensional workspace, each with an initial starting configuration (e.g. position and orientation) and a desired goal configuration, determine the path each agent should take to reach its goal, while avoiding collisions with obstacles and other agents in the workspace. More formally, let $\mathcal{A}$ be an agent in a static workspace $\mathcal{W} = \mathbb{R}^k$, where $k = 2$ or $k = 3$. The workspace is populated with obstacles. A configuration $\mathbf{q}$ is a complete specification of the location of every point on the agent geometry. The configuration space $\mathbf{C}$ represents the set of all the possible configurations of $\mathcal{A}$ with respect to $\mathcal{W}$. Let $\mathcal{O} \subset \mathcal{W}$ represent the region within the workspace populated by obstacles. Let the closed set $\mathcal{A}(\mathbf{q}) \subset \mathcal{W}$ denote the set of points occupied by the agent when it is in the configuration $\mathbf{q} \in \mathbf{C}$. Then,

the *C-space obstacle region*, C_{obs}, is defined as:

$$C_{obs} = \{\mathbf{q} \in \mathbf{C} | \mathcal{A}(\mathbf{q}) \cap \mathcal{O} \neq 0\}.$$

The set of configurations that avoid collision (called the *free space*) is:

$$C_{free} = C / \ C_{obs}.$$

A free path between two obstacle-free configurations c_{init} and c_{goal} is a continuous map:

$$\tau[0,1] \rightarrow C_{free}$$

such that $\tau(0) = c_{init}$ and $\tau(1) = c_{goal}$.

For a team of m agents, define a state space that considers the configurations of all the agents simultaneously:

$$X = C^1 \times C^2 \times \cdots \times C^m.$$

Note that the dimension of X is N, where $N = \sum_{i=1}^{m} dim(C^i)$. The C-space obstacle region must now be redefined as a combination of the configurations leading to an agent-obstacle collision, together with the configurations leading to agent-agent collision. The subset of X corresponding to agent $\mathcal{A}^i$ in collision with the obstacle region, $\mathcal{O}$, is

$$X_{obs}^i = \left\{x \in X | \mathcal{A}^i(\mathbf{q}^i) \cap \mathcal{O} \neq 0\right\}. \tag{4.3.1}$$

The subset of X corresponding to agent $\mathcal{A}^i$ in collision with agent $\mathcal{A}^j$ is

$$X_{obs}^{ij} = \left\{ x \in X | \mathcal{A}^i(\mathbf{q}^i) \cap \mathcal{A}^j(\mathbf{q}^j) \neq 0 \right\}. \tag{4.3.2}$$

The obstacle region in X is then defined as the combination of Equations 4.3.1 and 4.3.2, resulting in

$$X_{obs} = \left(\bigcup_{i=1}^{m} X_{obs}^i \right) \bigcup \left(\bigcup_{i,j,i \neq j} X_{obs}^{ij} \right). \tag{4.3.3}$$

With these definitions, the planning process for multi-agent systems treats X the same as C, and X_{obs} the same as C_{obs}, where c_{init} represents the starting configurations of all the agents, and c_{goal} represents the desired goal configurations of all the agents.

Typically, optimization criteria guide the choice of a particular solution from an infinite number of possible solutions. Example criteria include minimized total path lengths, minimized time to reach goals, and minimized energy used to reach goals. Additional constraints can introduce more complexity in the planning process, such as navigational restrictions on the agents (e.g., maximum slope restrictions, inability to traverse rocky terrain, etc.), or the need for multiple agents to move in tandem with each other (e.g., a formation of agents moving over uneven terrain). Since the general optimal motion planning problem for multiple moving objects is computationally intractable (specifically, PSPACE-hard [50]), most approaches relax the requirement for global optimality, and instead seek to locally optimize portions of the path planning problem.

4.3.2 Research Questions

The previous chapters review methods and techniques that are used or are highly related to the research in this PhD study. Each of the works carried out in this PhD

study is either an improvement made on previous studies reviewed in this chapter or an existing method integrated in a new way to achieve better results.

As a result of the reviews presented in chapter 2 and chapter 3, several research questions related to multi-agent collision avoidance can be formulated as follows:

1. Can we investigate and summarize all possible collision types in multi-agent systems? (Chapter 5)

2. Can we solve a deadlock situation in a cooperative way in a tunnel-like environment? (Chapter 6)

3. How can paths be planned for a moving goal in swarm robotic group formations? (Chapter 7)

4. How can a large number of agents autonomously and equally deconflict paths in dynamic and uncertain environments? (Chapter 8)

5. Can a large number of homogeneous autonomous agents extend to heterogeneous agents with variable size and speed? (Chapter 9)

The first two questions in particular are the foundation questions, since a clear understanding is required of what it is exactly that agents and robots must avoid. As noted in the literature reviews, there is surprisingly little work on this foundation issue. The fourth and fifth questions can only be tackled if an answer is provided to the first three questions. All of these questions will be answered in the rest of the thesis.

4.4 Proposed Method

4.4.1 Limitations of Previous Methods

The disadvantage of most of the previous decentralized collision avoidance approaches is that each method is restricted to a global view or requires communication between agents. In addition, none of them can take advantage of local view, and there is non-priority and no communication in a shared environment. This thesis claims that by adopting a nature-inspired approach, these disadvantages can be overcome.

4.4.2 An Ideal Collision Avoidance Method

An ideal collision avoidance method is: decentralized (no central control); distributed (each agent adopts the same procedure); localized (each agent has their own local view and does not depend on a centrally held data structure that gives a global view of the environment); non-communicative; reactive and autonomous (each agent is responsible for its own collision avoidance maneuver). In this study, inspiration for avoiding collisions comes from the use of roundabouts for resolving potential vehicle collisions at road intersections. When a vehicle approaches a roundabout, drivers adopt a specific set of autonomous but shared procedures for negotiating the roundabout to continue on their routes while avoiding collision with other vehicles already on the roundabout. Drivers do not require a central command and control structure or communication with each other to avoid collision. The idea adopted in the thesis is employing a temporary roundabout such that each agent independently locates in the possible collision position when two agents realized they may be on a collision course. The two agents then adopt shared but non-communicated procedures for negotiating

the roundabout and thereby avoid colliding with each other until they can resume their desired path.

While roundabouts are concrete in the real traffic world, the temporary roundabouts used here by agents are virtual: they do not actually exist in the configuration space but do exist temporarily on the paths of each agent involved in a possible collision before disappearing from each agent's paths once collision is avoided. No other agent sees this roundabout. Because the configuration space consists of rectangular grids, the virtual roundabout necessarily becomes straight-lined and rectangular, hence it is called a 'rectabout'. Moreover, real roundabouts can vary in size depending on various parameters such as approach speed, one-lane or multi-lane, number of roads leading to that intersection, etc.

In order to design an effective collision avoidance model, two points should be taken into consideration: (1) no communication while avoiding collisions since true autonomy requires no central coordinator and no communication between agents; (2) how to ensure that agents resume their paths after collision avoidance to still reach their goals effectively and efficiently.

4.4.3 The Idea of MER-based Collision Avoidance

An ideal roundabout collision avoidance system has to satisfy the following principles:

1. Flexibility in size: since the degree of collision risk depends on the distance between the two agents, the system should have the ability to shift its scale according to the distance of the nearby agents.

2. Orientation: the topology of the roundabout must be such that it is correctly oriented to deal with conflict. That is, the location and orientation of the

roundabout must be able to deal with two robots approaching each other from any angle and not assume fixed entry and/or fixed exit points.

3. System adaptability: the system can be used for resolving any type of possible collision for multi-agent systems.

4. System independence: the system should not be restricted to deal with a predetermined number of agents, a fixed local view or a predetermined and fixed configuration space.

In this thesis the Minimum Enclosing Rectangle (MER) is used to support these principles. According to Das et al. [23], given a set of points $P = \{p_1, p_2, ..., p_k\}$ with $p_k \in \Re^2$, the minimum enclosing square (or rectangle) of P is the smallest square (or rectangle) that contains all points of P. For the purposes of this research, the smallest square or rectangle is defined as the smallest rectangle that contains a given number k such that $\frac{n}{2} < k \leq n$ of x-consecutive points in a set of n points in the plane. The problem of computing *k-square* and *k-rectangle* has been investigated since 1991 (for a review, see [1, 31, 115, 23, 79]). MER has been applied in various areas, such as pattern recognition [96], facility location [29], similarity search [88, 24] and collision detection [75]. In order to classify the *k-square* with respect to the number of points η present on its boundary, Das et al. [23] investigated all the different possibilities of *k-squares*. As a result, no *k-square* is possible with $\eta = 0$ or 1. The only possibility with $\eta = 2$ is that the two points appear at the two diagonally opposite corners of the corresponding *k-squares*. Nevertheless, MER has potential to be used to address multiple points at the same time for collision avoidance. This could be one direction of future work. In this study, $k = \eta = 2$ is the MER or MES that the robots are

searching for, as shown in Figure 4.2.

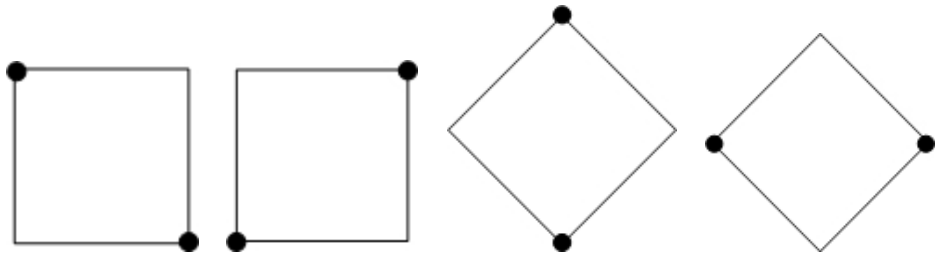


Figure 4.2: MER of $\eta = 2$, with dots representing the position of the two agents. The orientation of the rectabout can differ according to the local view.

In this study, MER as a potential roundabout representation method is considered for the following reasons. First, MER encloses agent positions in the smallest rectangle or square irrespective of the distance from the other agent, and the MER itself is flexible in size. Second, MER can find the suitable location and orientation to enclose agent positions at the two diagonally opposite corners $\eta = 2$. Third, MER is an independent system representation method. Because the enclosed agents in MER are not constrained by specific synchronous, symmetric and communication-based restrictions, these agents are fully autonomous in their moving direction along the paths given by the roundabout.

The proposed method builds on an implicit assumption that other agents make similar collision avoidance reasoning via MER. That is, knowledge of MER is shared by all agents. It consists of two components: Minimal Predicted Distance (MPD) detection and MER rectabout collision avoidance algorithm. The MPD is a metric inspired by real human pedestrian collision avoidance behaviour (for a review, see [93, 92] and more details below). As far as we are aware, this is the first time that MPD

has been used for addressing collision problems in multi-agent systems. The MER-based rectabout collision avoidance algorithm is a pairwise approach which computes and re-plans agents' moving direction by following a 'keep left' rule at the rectabout. This rule can be changed if necessary to keep right. The proposed collision avoidance method is demonstrated for multi-agent systems using different types of collisions investigated in [76].

4.5 Design of Experimental Methods

In this research work, the algorithms are applied to distributed multi-agent systems. At each time step, an agent has potentially 9 legal actions, irrespective of the local view schema adopted by agents. This will be described later in the thesis. Each of these legal actions is a solution to the constraint satisfaction problem in which each agent must be assigned a move from {E,S,W,N,NE,SE,SW,NW,wait}, and there are constraints between sets of legal moves. For example, these legal moves must not lead to two agents colliding. With respect to preventing collisions, the algorithms presented in the work are intended to cope with local view collisions in various situations, such as crowded environments and tunnel-like environments. We adopt both simulated agents and robots for the experiments, since our robots have the capacity to plan and resolve conflicts between themselves, just like autonomous agents. We employ both implicit communication methods and explicit communication methods for robots to handle all the data which are transmitted via the network system, shown in Figure 4.3.

For multi-agent and motion coordination rehearsal, a four-step experiment on multi-agents is planned as follows,

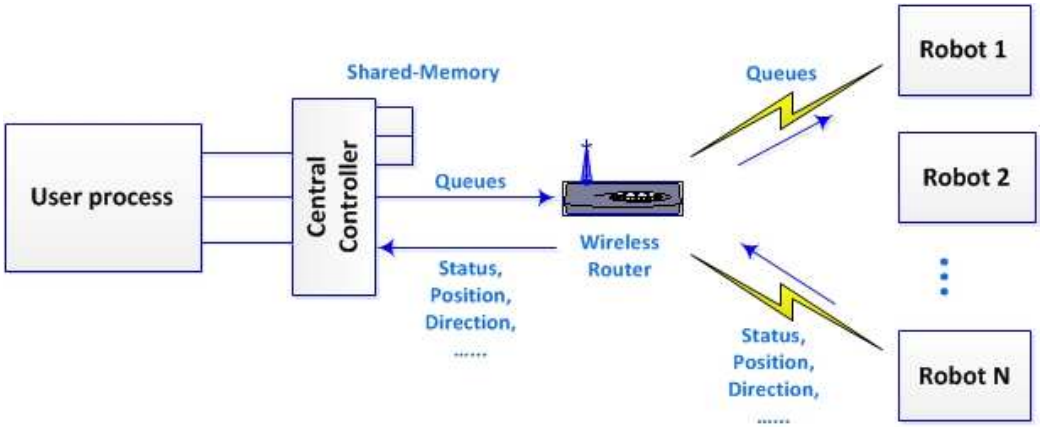


Figure 4.3: Communication architecture in multi-agent systems.

1. Stage 1 - centralized coupled system - experiments with collision types [73, 76] in multi-agent / robot systems. All agents move to their targets synchronously without collisions, including encountering fixed and dynamic obstacles through a centralized coupled approach.

2. Stage 2 - centralized decoupled system - two experiments are conducted in this stage, one with the adaptability of motion coordination and automatic obstacle avoidance behaviour. The other is with the agent swarm from an initial random start position which must coordinate their behaviours to form a shape (e.g. circle, square) such that each agent is equidistant from its neighbours.

3. Stage 3 - autonomous homogeneous system - multiple homogeneous agents move from start position to goal position with local view through a decentralized approach. Communication is not allowed between each other. All start and goal positions are randomly assigned.

4. Stage 4 - autonomous heterogeneous system - heterogeneous agents are applied to the system, achieving the goal of Stage 3.

Figure 4.4 shows the journey of this PhD research.

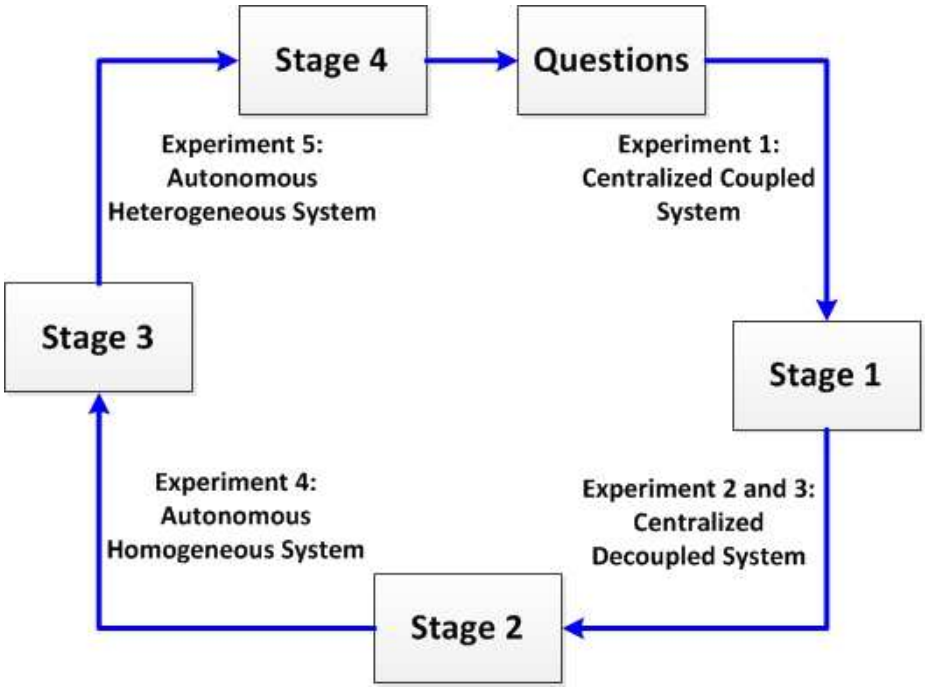


Figure 4.4: The journey of this PhD research.

4.6 Analysis of Results and Validation

The analysis and evaluation of this research is carried out by using a centralized multi-agent pathfinding simulator, a decentralized multi-agent system simulator and Matlab tools. Figure 4.5 and Figure 4.6 illustrate the experimental environment for centralized and decentralized multi-agent simulators, respectively. With centralized simulator, agents settings (number of agents, start and positions, etc), group settings and environmental settings can be interacted with user. With decentralized simulator, it is very similar to the centralized one, but the differences are approach setting (A* or A* with other algorithms) and rules setting (keep left or keep right for different

traffic controls).

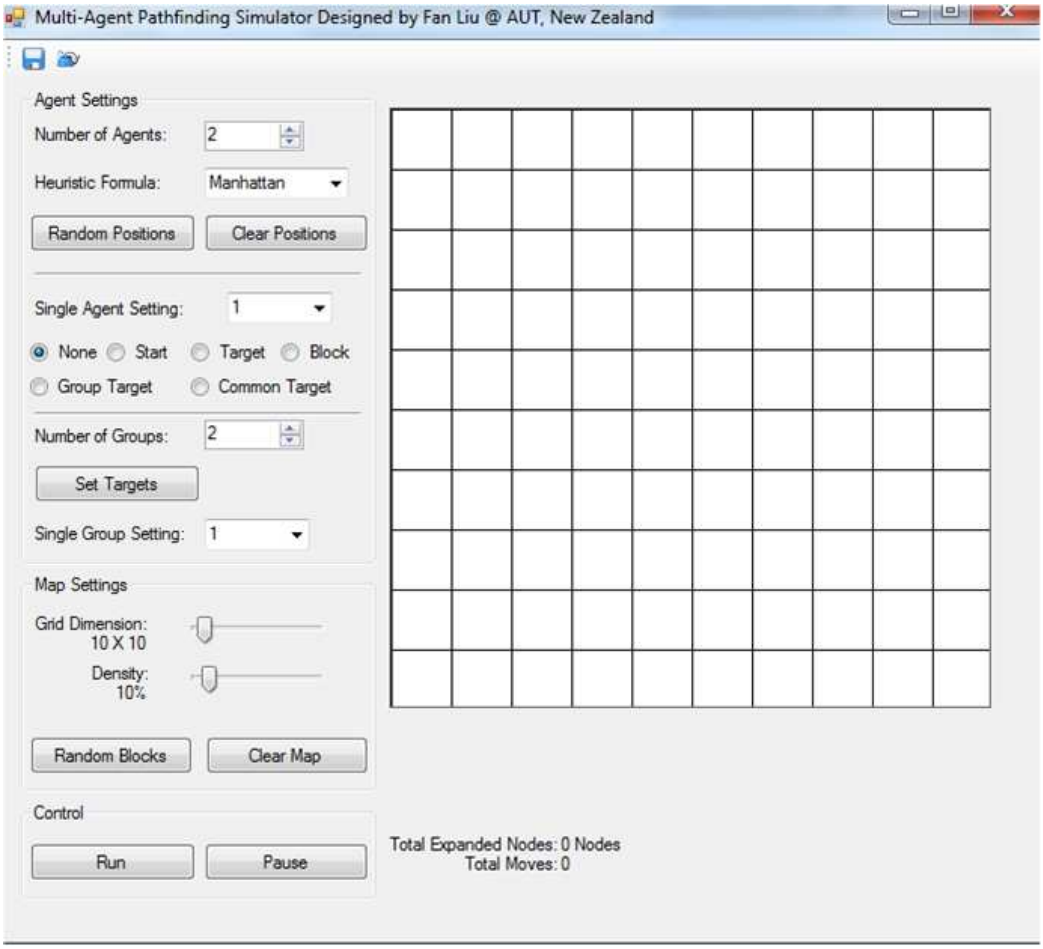


Figure 4.5: Illustration of experimental environment for centralized multi-agent simulator.

4.7 Review and Evaluation of Output Reports

The proposed approach will be tested on real robots and in extensive simulation runs to answer the following questions. (1) Practicability: Is the proposed approach relevant and applicable to real robot systems? (2) Solvability (Completeness): Does

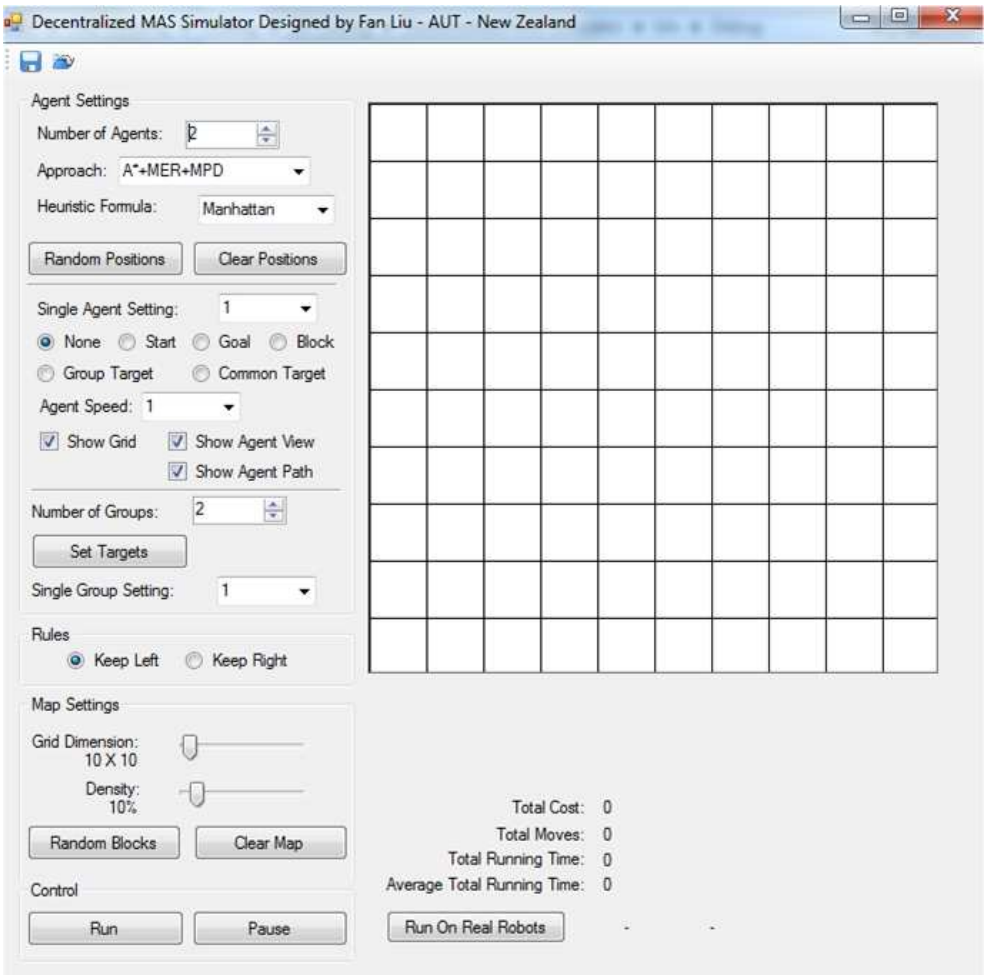


Figure 4.6: Illustration of experimental environment for decentralized multi-agent simulator.

the proposed approach succeed in finding valid collision-free multi-robot paths if one exists? All experiments are carried out using different configuration environments. We apply the proposed method to the two robot systems with deadloop and collision conditions in [73, 76]. The control architecture is through networked robotics to allow mobile robots to communicate with each other via Wi-Fi and through a coordination control laptop that relays path and obstacle information to all mobile robots. In simulated validation, the robustness of collision avoidance needs to be evaluated on:

(A) the capability of agents' positions, i.e. the system is not restricted to any synchronous, symmetric and communication-based maneuvers; (B) the adaptability of collision avoidance in case of deadlock; and (C) the scalability of collision avoidance to increasingly larger numbers of agents.

4.8 Summary

This chapter presents a scientific research methodology in the form of sequence analysis. It is the best possible way to focus the research process and organize the research by formulating and defining a research problem and drawing conclusions that reflect the real world.

The research methodology identified in this chapter makes a crucial assumption, which is that our contribution of inspiration from nature will be introduced not from the very beginning of the research but at a point in the research where it is most desirable to do so. As can be seen in Chapters 2 and 3, there are already a lot of knowledge and information concerning formal path-based search approaches and collision avoidance approaches in the standard literature. While these classical approaches are well understood and, in many cases, formally grounded, they assume a compromise of the ideal scenarios identified in the Introduction chapter: these approaches assume a mixture of centralized control, communication, priority and a global data structure. That does not mean that we have to start again with a blank page with a nature-inspired approach. That would be throwing away too much good work that already exists. The task for this thesis is to take those aspects of searching, planning and collision avoidance that are well understood and relevant to our research, and then to see how we can add to them with inspiration from nature at that point in the research when

it is most appropriate to do so.

In conjunction with the research questions above is also a suggested method: see how far we can get with current techniques and extending them where necessary to make them more suitable for collision avoidance. We believe that we have gone as far as we can with these classical methods and see how inspiration from nature can lead to new insights as well as possibly novel solutions. These covers the perspective of the two scenarios introduced in Chapter 1, for autonomous, non-centralized, non-global, non-communicative collision avoidance.

We therefore need to start with what is currently known, or not known, about collision avoidance which leads to the encounter of our first problem. As was shown in the literature reviews, despite all the work done so far in collision avoidance, there is actually very little formally stated on what a collision is and how to recognize a collision.

The next chapter investigates all possible collision types in multi-agent systems. A novel extension to the standard A* algorithm using centralized coupled approach is presented and experimental results follow which relate to research question 1.

Chapter 5

Super A* based on Collision Type

The aim of this chapter is to address the first research question identified in the Research Methodology chapter: Can we investigate and summarize all possible collision types in multi-agent systems? The purpose of this chapter is to evaluate our collision avoidance classification using a path-searching method that is well understood by classical researchers so that our identification of collision types is kept separate from questions about detecting and avoiding them. If a standard, well-understood approach can be shown to work with our collision types, that will count as some evidence that our classification may be correct. Nevertheless, changes will need to be made to the standard approach to cope with the collision types identified. In this chapter, we use the standard A algorithm as our basic, well-understood method to demonstrate that, with suitable and novel extensions, it can implement avoidance strategies based on our collision types.*

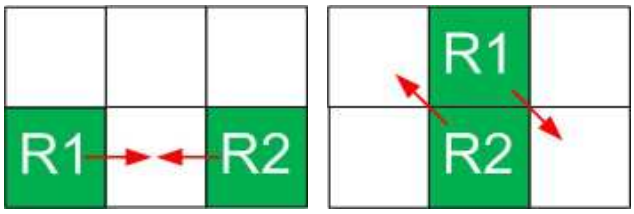
This chapter will therefore investigate and summarize all possible collision types in multi-agent systems. Section 1 describes collision avoidance issues in existing approaches. Section 2 defines a number of collision types for multiple agents and, in particular, between pairs of agents that need to be handled by any real-time collision

avoidance systems. In Section 3 a novel extension to the standard A algorithm is presented (Super A*) that solves the problem of these collision types by using dynamic real time monitoring and iterative move-evaluate-move cycles. The evaluation of the Super A* algorithm is presented in Section 4. A discussion on Super A* performance is given in Section 5.*

5.1 Introduction

The area of dynamic multi-agent systems has many unresolved interaction problems. The goal of this chapter is to describe an alternative approach to collision avoidance that is intended for use in real-time situations. To achieve this, it is necessary to identify all possible collision types in a multi-agent system.

Existing multi-agent pathfinding approaches ignore sideswipe collisions among agents (i.e., only consider the collision where two agents try to occupy the same node during the same time-step) [54, 120, 126], and allow diagonal moves between two adjacent nodes (e.g., Figure 5.1(b)). However, in many real world applications, sideswipe collisions may also block agents' movements or cause deadlocks. For example, as shown in Figure 5.1, if the size of two agents is as big as the grid size they occupy, collisions will happen not only between agents R1 and R2 in the situation depicted in Figure 5.1(a), but also in Figure 5.1(b), which is typically not considered as a collision in existing multi-agent systems.



(a) (b)

Figure 5.1: R1 and R2 represents robot 1 and robot 2, respectively. (a) Occupy the same position. (b) Sideswipe collision.

5.2 Deadloop and Collision Types

To overcome the limitation depicted in Figure 5.1(b), we investigate all possible collision scenarios in a multi-agent system (the speed / velocity of agents is taken into consideration when describing these collisions) when agents are moving, and identify one deadloop type and five collision types. We show that all possible scenarios that may hinder an agent's planned motion in a two-dimensional space can be covered by these collision / deadloop types (with symmetry). The scenario that may cause a deadloop situation in a multi-agent system, on the other hand, is depicted in Figure 5.2. The five collision types are head-on, front sideswipe, rear sideswipe, front-end swipe and front-end sideswipe, which are illustrated in Figure 5.3(a) to (e), respectively. Front sideswipe (Figure 5.3(b)) and rear sideswipe (Figure 5.3(c)) can occur only on diagonal moves for both agents.

In this research, we model the motion space of agents as a grid map, and assume that an agent can be as big as, or smaller than, the cell. The Super A* algorithm, which will be introduced in the following section, can handle all these collision / deadloop situations, and allow agents to reach their destinations.

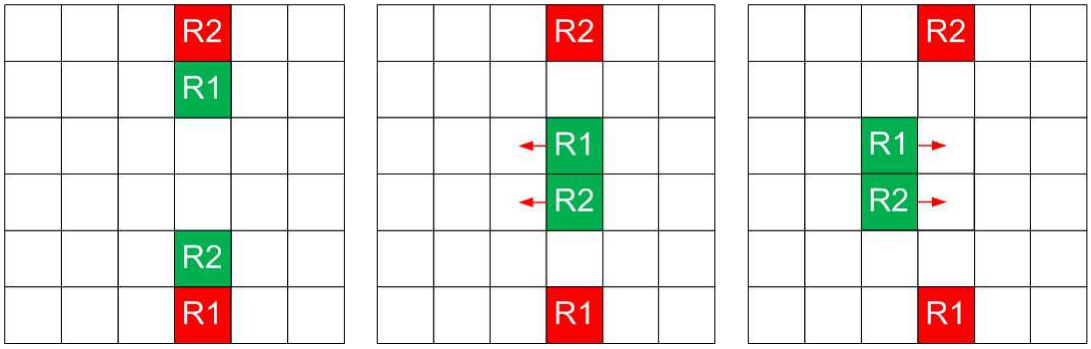


Figure 5.2: Illustration of deadloop. The green squares and the red squares are the agent positions (R1, R2) and the goal positions (G1, G2) for two agents, respectively. R1 and R2 are agent 1 and agent 2. (a) The initial position for two agents. (b) and (c) The deadloop condition is encountered and repeated in-between (b) and (c) infinitely as each agent makes a move that mirrors the other agent.

5.3 Prioritized A*-based Path Planning and Step-Forward Path Coordination

The A* search algorithm [89] is widely used in computing cost-optimal paths for individual agents. It uses branch and bound supplemented by heuristic information and pruning of redundant paths to find the time or length optimal collision-free path from start point to goal point [46]. In this work, we assume that the environment, or configuration space, is simulated by a discrete occupancy grid map. This assumption is widely used in the robotics literature. An occupancy grid consists of a grid of equally spaced cells of arbitrary size. Here, we use a two-dimensional space, i.e., the 'environmental matrix', to represent the physical environment. Each cell in the matrix can be empty, occupied by a static obstacle, or occupied by an agent. Paths for agents are represented as moves through or over grids in the two-dimensional space.

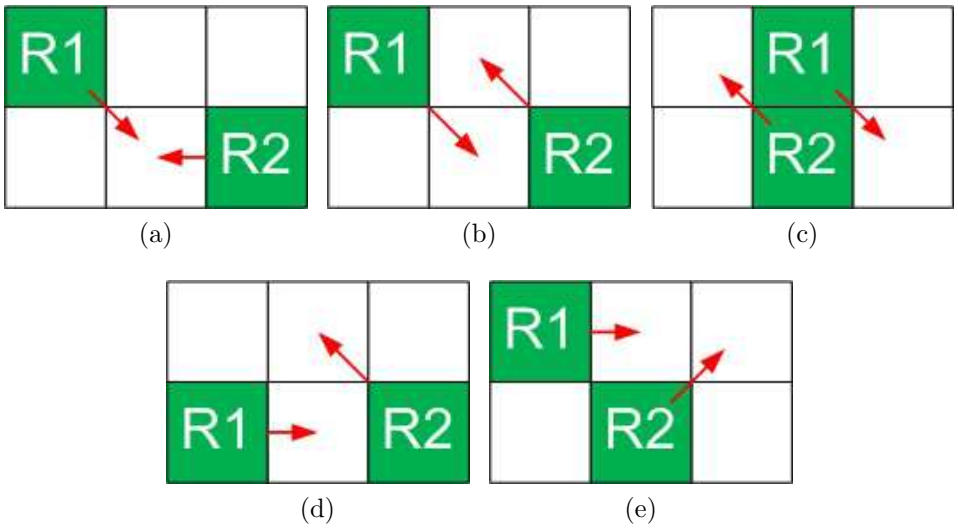
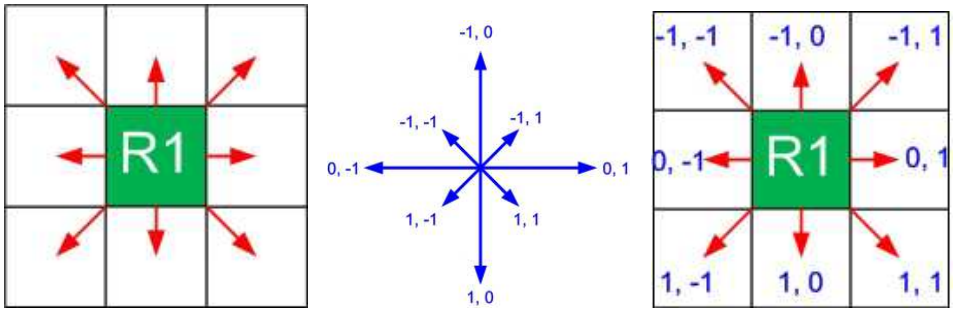


Figure 5.3: Illustration of 5 collision types. (a) Head-On. (b) Front Sideswipe. (c) Rear Sideswipe. (d) Front-End Swipe. (e) Front-End Sideswipe.

Given two specific locations, i.e., the initial position (x, y) and the goal position (x^*, y^*), the objective of A* is to estimate the lowest cost path from position (x, y) to (x^*, y^*). The gone cost G and heuristic cost H are involved to calculate the total cost F ($F = G + H$), which is used to guide the search (the path with least F is explored next). Figure 5.4 illustrates the coordination scheme for moves in the simulation environment. There are nine cells which represent eight possible single move directions (nodes) for R1, which is located in the center of this space. The gone cost is the actual cost moving from the currently occupied cell (the central node) to one of the 8 neighboring nodes, which then becomes the newly occupied cell. Heuristic cost is the estimated cost remaining from the newly occupied cell to the goal position. Here, we use the Manhattan formula as the method to estimate distance remaining (which takes no account of any obstacles), given by $\mathcal{C} \cdot (||x^{newnode} - x^*|| + ||y^{newnode} - y^*||)$, where $\mathcal{C} > 0$ is chosen as the Heuristic Coefficient. The optimal path is the path

with the lowest F cost ($F = G, H = 0$), and can be found by using the heuristic information with no 'open' paths left (i.e. paths that may reach the goal with less actual cost).



(a) (b) (c)

Figure 5.4: The coordination scheme for possible moves. (a) 8 possible moving directions for Agent 1. (b) The coordination scheme. (c) The possible move of R1 based on the coordination scheme.

5.3.1 Prioritized A*-based Path Planning

Super A* is a new algorithm for solving the multi-agent path planning problem. The major issues for re-planning relate to the deadloop and the five collision types defined above. In collision avoidance, determination of priorities is a key factor that may result in different re-planning outcomes. When dealing with agents pausing to allow other agents to move, priority can be assigned to agents based on a static factor, such as the order in which they are placed on the grid [73], or based on some dynamic factors, such as which agent is closer to its target or which agent reaches a potential collision point first. Priority is only introduced when dealing with possible collisions between individual agents. It does not affect the optimal paths of other agents. In this way, global and local optimal solutions are taken into account.

The basic ideas of the Super A* algorithm are summarized in Table 5.1 and Algorithm 1.

Table 5.1: Super A* Notations

Notation	Descriptions
n_0	Starting node
n	Ending node
L	Label of agent
pn	Parent node
N_{open}	OPEN-list of node
N_{close}	CLOSE-list of node
n_{new}	New node of current parent node
G	Gone cost value
G_{new}	New gone cost value

Super A* is a fixed priority scheme and there is no calculation for priority. This is a coupled procedure that each agent runs Super A* one by one in order. The collision types are built into each agent and checked at stage 10 of the Super A* algorithm which works in an intelligent grid that describes the position of every agent, its start and end path nodes, and stores the paths calculated from the previous Super A* run. Before generating a full path in the next run of Super A* (stages 15-22), the agents check for possible collision types and deadloop on the various paths in the intelligent grid, where agents employ 'flagged' if two or more agents are going to collide. If collision is flagged, the agents find an alternative route avoiding the collision, but this alternative route is also checked at the next run of Super A*. The partial path of the new node does not exist, because in A* algorithm the new node is one of 8 neighbour nodes for calculating the heuristic cost from the new node to goal node.

Algorithm 1 Super A* Algorithm

Input: Input two nodes n_0, n and L
Output: Output a set of nodes $N_{close}, n_i \in N_{close}$

$N_{open} \leftarrow n_0$, $N_{close} \leftarrow \emptyset$, flagged $\leftarrow$ **false**
loop
  $pn \leftarrow$ Compute the lowest cost of node, in N_{open}
  if $pn = n$ **then**
    $N_{close} \leftarrow N_{close} \cup \{pn\}$
    Return N_{close}
  else
    for all neighbours n_{new} **do**
      if L accords with a fixed priority scheme **then**
        flagged $\leftarrow$ Check Deadloop and five Collision types
      end if
      if flagged is **true then**
        Continue Loop
      end if
      if ($n_{new} \in N_{open}$ or $n_{new} \in N_{close}$) and $G_{new} <$ current G **then**
        $G \leftarrow G_{new}$
        Continue Loop
      end if
      $N_{open} \leftarrow N_{open} \cup n_{new}$
    end for
  end if
  $N_{close} \leftarrow N_{close} \cup \{pn\}$
end loop

5.3.2 Step-Forward Path Coordination

Given a common space, a start node and a goal node set for each agent, a path is computed for each agent that avoids collisions with fixed obstacles. As noted earlier, a major limitation of the A* algorithm is that it can find the optimal path in a static environment but not in dynamic environments where A* is started again with the new configuration. Therefore, in [73], we extended the A* algorithm and propose the Super A* algorithm which can handle dynamic obstacles in real-time (i.e. without restart). Given two agents R1 and R2, R1 can become a dynamic obstacle in relation to R2, and vice versa if collision is possible between them (Figure 5.2). The Super A* algorithm consists of redesigning A* as a real-time algorithm for handling

dynamic environments through a step-forward dynamic moving approach ('move-evaluate-move'). The following procedure is repeated until the goals are achieved by all agents. The extension to A* is as follows:

1. The agent with Label 1 computes its optimal path using Super A*. Then, pass the planned path to the agent with Label 2.

2. Each agent moves one step following the established path.

3. Each agent evaluates the current environment. If the goal is achieved, then the algorithm is stopped for that agent. Otherwise, the algorithm runs again until all agents achieve their goal.

A single agent keeps updating its path by using the most recent information available, including the positions of other agents and their planned next moves. Deadloop and five other collision types are identified one step ahead. For these situations, the Super A* algorithm applies a single and fixed priority scheme based on the order in which agents are placed on the grid one step ahead of the collision for preventing collisions between agents, and then agents are returned to their optimal paths as quickly as possible.

5.4 Evaluation

We have conducted both real robot and simulator-based simulations. Through these simulations, we try to evaluate the following three aspects of the proposed strategy: (1) Practicability: is the strategy relevant and applicable to real robot systems? (2)

Solvability: can the strategy find valid collision-free multi-robot paths? (3) Optimality: is the strategy able to generate the best paths despite collision-avoidance behaviour? Experiments were carried out using different configuration environments. The proposed method is applied to a two-robot system with the deadloop and collision conditions described in Section 5.2. Video links to demonstration are provided at the end of the chapter.

5.4.1 An Example with Real Robots

The goal of the first experiment is to demonstrate the applicability of our approach to real robot systems. This experiment was carried out using the Rovio robots of WowWee Technologies. The task of the robots is to find the optimal / shortest path and move from their initial positions to their goal positions without collision. Figure 5.5 shows one experimental robot and one experimental physical space, which is a 5 by 5 grid with each grid measuring 50cm by 50cm. In Figure 5.6(a), the robots are deployed on two sides of the field and have to move to their goal positions on the other side using Super A* and avoiding deadloop and collision. Figure 5.6(b) shows the two robots rotating their directions from the initial situation. As can be seen in Figure 5.6(c), the paths of the robots are changed in order to avoid collisions with each other. For the optimal paths computation, robot 1 and robot 2 both have to take a detour around each other. At the end, the two robots achieve their goal position shown in Figure 5.6(d). This is the optimal resolution to this possible collision without introducing a sideways move or a front end swipe or sideswipe.

(a) (b)

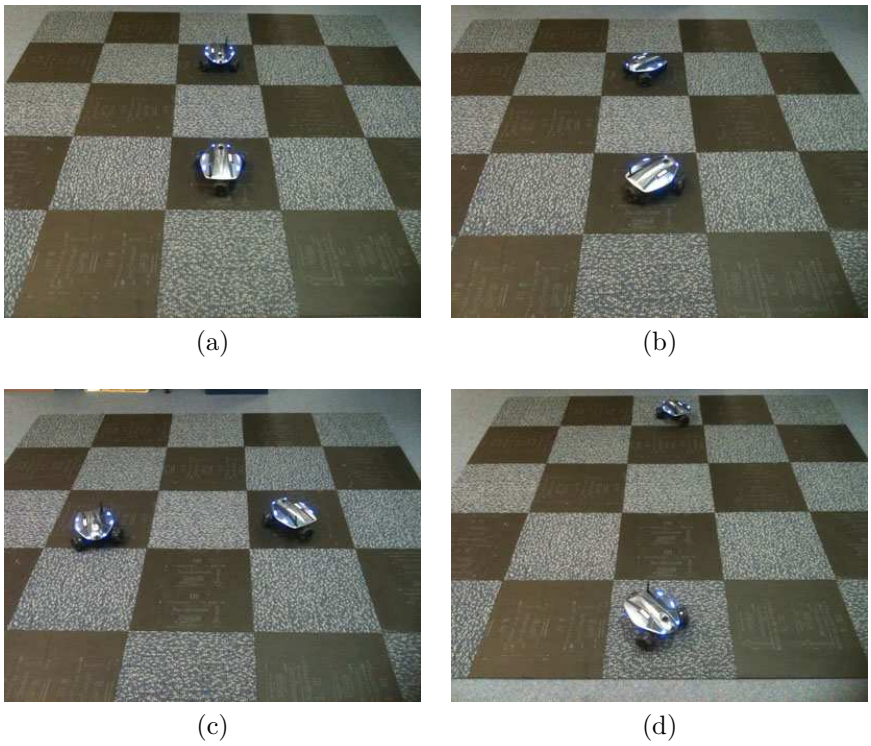
Figure 5.5: These two pictures show the robot used in the experiment as (a) and configuration space in real world as (b).

5.4.2 Simulation Experiments

In the simulation experiments using a 10 by 10 grid, we analyze the conditions with collision and without collision. The 'without collision' condition may arise when agents need to coordinate their actions and synchronize their movements to ensure a proper sequence is achieved.

Priority with collision

The first set of experiments was performed to investigate the effects of the priority scheme. In Figure 5.7(a), the blue squares represent the calculated paths of the agents, and the orange squares are the potential collision risk where the paths of the two agents overlap. Without a priority scheme, it can be seen in Figure 5.7(b) that agent 1 and agent 2 both find sub-optimal paths to avoid collision. But with priority, optimal collision avoidance is achieved. In Figure 5.7(c), R2 lets R1 take the optimal path and R2 selects an avoidance strategy that allows it to return to the optimal path after collision is avoided.



(a) (b)

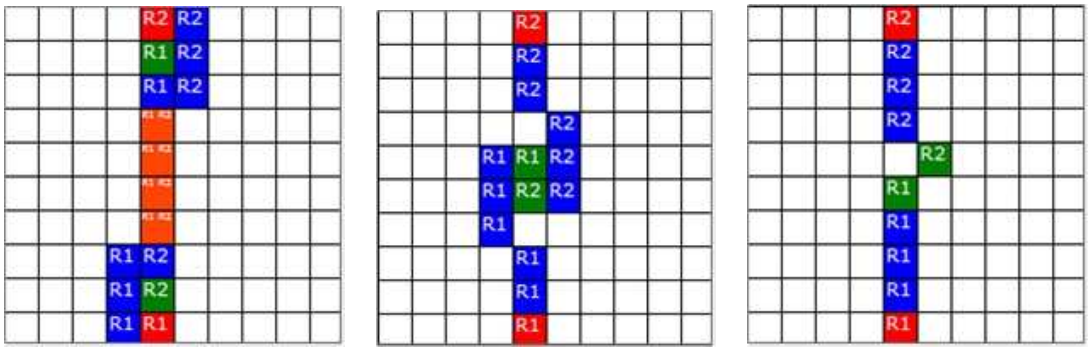
(c) (d)

Figure 5.6: An application example with the Rovios of the WowWee Technologies: (a) shows the initial situation of two robots; (b) shows the two robots rotating in their cells to avoid collision; (c) depicts the two robots passing each other with no collision; and (d) shows the two robots in their goal positions.

Priority without collision

The final experiment in this section is designed to illustrate that the priority scheme is flexible and can be used to find the optimal / shortest paths for multiple agents in the presence of fixed obstacles rather than collisions. Figure 5.8(a) demonstrates the environment of two agents needing to pass through a tunnel to reach their goals.

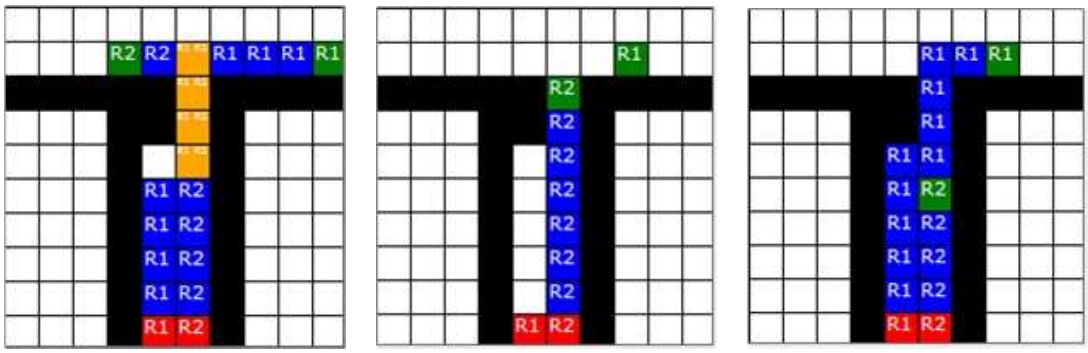
In Figure 5.8(b), R2 gets to the tunnel first, so R2 has the priority to go through the tunnel. R1 must stop and wait until R2 passes. Then, R1 follows, as shown in Figure 5.8(c). This is an example of allocating priority based on time, in this case,



(a) (b) (c)

Figure 5.7: Illustration of the priority with collision condition: (a) shows the original calculated paths; (b) shows sub-optimal collision avoidance without priority; and (c) shows Super A* (with priority) and optimal collision avoidance.

which robot reached a potential collision point first. The reason why R1 has to wait by the entrance of the tunnel until R2 has passed through the tunnel completely is that, in Super A*, R2 and other obstacles represent a dynamic, unavoidable and unpredictable obstacle. That is, because there is no other path available to R1, and because there is no certainty that R2 may not change direction in a subsequent move. There is no point in R1 moving to be right by the tunnel entrance until a clear path is found. Also, a 'fixed' obstacle, such as part of the tunnel at the extreme right of the space, may be removed due to the changing and dynamic nature of the configuration space, thereby allowing R1 another route to its goal. If R1 moves to the mouth of the tunnel before R2 has exited the tunnel, R1's path to its goal, will be non-optimal, should a gap open up at the extreme right of the configuration space.



(a) (b) (c)

Figure 5.8: Illustration of priority without collision condition. Once R2 which has priority because it reaches the tunnel first, is clear of the possible collision nodes in the tunnel, R1 can complete its moves.

5.5 Discussion

The aim of path finding algorithms is to find the shortest / optimal path from the start position to the goal position. The application of optimal path techniques to robotics and multiple agents in particular has a long history [65], as does the application of A* in multiple constrained search spaces [77]. Despite much early work in multiple agent planning, there is no clear understanding as to whether collision avoidance and replanning should be included explicitly in such planning algorithms (potentially slowing down the algorithms) or whether collision avoidance should be left to agents to negotiate at the local level. The Super A* algorithm assumes that replanning should be built into A*, taking into account the many different types of collisions possible, so that global requirements of optimality can be addressed. Due to major increases in computer speed since the early days of robotic planning, the major bottleneck now can be expected to lie in inter-robotic communications. Leaving agents to deal with collision avoidance locally is subject to the 'horizon effect', when a new danger

presents itself that could lead to robots taking non-optimal replanned routes, which could further increase the risk of collisions. The proposed algorithm has been shown capable of dealing with both static and dynamic obstacles. Each agent is able to find the optimal path or, if it has to move off the path to avoid collision, to resume its optimal path. The extensions to standard A* involve the use of a step-by-step recalculation once the full paths have initially been calculated. Agents are allowed to progress on their optimal paths until collision is detected one step ahead or an obstacle is encountered. This delay in collision detection allows the system to be in an optimal state for as long as possible and to take into account obstacles that appear from over the horizon. Once the possibility of collision is detected, the appropriate avoidance behaviour is taken and the agents return to their optimal paths as quickly as possible. The extensions to A* rely on a networked robotics architecture that allows collision avoidance to be built explicitly into the replanning process. However, there is still work to be done to ensure that collision and replanning in Super A* adhere to global system requirements of optimality, since the approach is 'optimistically optimal', i.e. we cannot yet prove that Super A* is optimal in avoiding collisions.

Finally, further experiments need to be conducted to evaluate the behaviour of Super A* in the context of state-space configurations of increasing complexity. The current estimate of complexity (node expansion) is:

$$l * (b - o) * \sum_{i=1}^{m} m$$

where l is the number of robots, b is the branching factor (8 possible directions in our case), o is the average number of obstacles at each stage of the expansion and, m is the number of total moves. So for 1, 2 and 5 robots, with 10 and 20 obstacles in

each case, this leads to 475, 1394 and 3319 predicted node expansions (10 obstacles), and 352, 1327 and 2766 predicted node expansions (20 obstacles), respectively. Over 10 experimental runs with random start and goal points, the average actual numbers of nodes expanded were 447, 1092 and 2401 (10 obstacles), and 381, 856 and 3511 (20 obstacles), respectively, indicating that our estimate of complexity is reasonably accurate.

5.6 Supporting Information

5.6.1 Video S1 Super A* in a Two Robot System

The Super A* algorithm is redesigned based on the traditional A* search algorithm. It applies a single and fixed priority scheme for preventing collisions between robots one step ahead of the collision and then returning the robots to their optimal paths as quickly as possible. A video regarding this algorithm ca be retrieved at: http://www.youtube.com/watch?v=Qxseu5P4ZlM (4.78 MB MOV)

5.6.2 Video S2 Super A* in Extra Simulation Demos

Three scenarios are simulated in a 10 by 10 grid. The first scenario demonstrates deadloop and all five collision types, with dynamic path replanning demonstrated. It can be seen that R1 lets R2 take the optimal path, and R1 selects an avoidance strategy that allows it to return to the optimal path after collision is avoided. In addition, the proposed strategy can also cater for a combination of possible collisions. For instance, if a dynamic change to one or more agents' goal states leads to a deadloop condition, the proposed strategy can resolve the problem effectively. The

second scenario shows the tunnel-like environment, where two agents need to pass through a tunnel to reach their goals. R1 gets to the tunnel first, so R1 gets the priority to go through the tunnel. This is an example of allocating priority based on time. That is, the proposed strategy not only considers the optimal path cost but also takes optimal time cost into account. Finally, the last scenario shows that, with randomly changing goals in real time, the proposed strategy is capable of avoiding collisions and returning to the planned optimal path for the new goal nodes. An environment is randomly generated on 50 by 50 grid for 20 and 50 robots, respectively, with 10% obstacle density. The video link is http://youtu.be/gEHRxpbD_LY.

5.7 Relation to Previous Work

As noted in the literature reviews, collision avoidance is a well studied area in multi-agent research. In [102], Parsons and Wooldridge introduce an agent conflict resolution strategy which adapts the concept of utility from classical decision theory and applied to electronic commerce. This adaptation resulted in two decentralized conflict resolution algorithms for multiple autonomous aerial vehicles usage [122, 104]. The first collision avoidance algorithm was an iterative peer-to-peer algorithm, where the two agents (aircraft) involved in a possible collision negotiate collision avoidance. The second algorithm used a multi-party approach, where a group of agents or aircraft together manage to avoid the collision so that the effects of a pair of aircraft avoiding a collision would not impact on other aircraft. A version of A* [89, 46], was used to implement the multiple-party collision avoidance strategy. The equivalents of distance remaining and actual costs of the route were used in the utility function to queue the aircraft in a collision avoidance data structure to ensure that a collision

avoidance strategy was devised for each aircraft. One advantage of the multiple-party algorithm is that path finding does not need to be restarted again after each collision avoidance. However, it cannot cope with changing goals of aircraft. In other words, aircraft will have fixed destination landing sites, whereas agents may have multiple sites to visit while undertaking a task. In addition, since aerial vehicles cannot stop in mid-air, collision avoidance between two or more agents requires the craft to move continuously. In the robotic domain it is possible for an agent to pause to give priority to another agent. This pause feature raises the question of how priority is to be calculated to determine which agent should pause and which should continue to move. In a true dynamic environment, the goal states can also change due to unforeseen events. Optimality in terms of finding the least cost route for all agents will also need to be reviewed to take into account temporary halts in path traversal. As far as we are aware, there has been no previous work on optimality in the A* algorithm that includes not just distance traveled and distance remaining but also the effects of temporary halts. While the A* algorithm can be employed to solve the problem of the optimal / shortest path in a centralized approach, it will have difficulties when anything changes in the environment that leads to recalculation. The A* algorithm can also be used in a decoupled approach as will be seen in the next chapter. If the agent can move freely regardless of other agents' positions, the problem can be easily separated into many local optimal path planning problems. However, agents cannot occupy the same position at the same time. Figure 6.1 shows a situation in which a collision will occur between two agents if each agent runs and executes its own A* as if in a single agent system.

D*, or dynamic A* [16], is another algorithm based on A* for real-time re-planning

in dynamic environments. D* generates an initial plan based on known information and then incrementally develops the plan as new information is discovered. However, explicit methods for avoiding collision are not incorporated into D* on the assumption that any potential collision between two agents can be resolved 'locally' between just these two agents. Further extensions to D* were incorporated into GRAMMPS [17], but not explicit collision avoidance mechanisms. Agents may need complex sensors and movement-effect predictors to ensure the effectiveness of D*.

RTA* and LRTA* [119] use a search with limited lookahead (look ahead to a specific depth), which it bases on minimax search, and calls minimin search, as costs should be minimized on all layers. The search is a modified Dijkstra search, or a BFS (Breadth First Search) with re-expansion. As with A*, a heuristic function is used with the search. From the limited lookahead search, a better heuristic value for the node is obtained and stored in a hash table containing improved heuristic values. This limited lookahead is also the reason why (L)RTA* maintains real-time properties for computational cost and memory usage. When looking up a heuristic value, this table is primarily consulted, and if that does not yield an answer, the regular heuristic function is consulted. The hash table serves to build an improved heuristic function for the domain. This type of search is non-exhaustive and thus does not guarantee that the agent will move on a path directly towards the goal. That is, it may fall into local minima of the heuristic function. However, thanks to the hash table of stored heuristic values, it will eventually work itself out of the minima and continue on a better path towards the goal. The main difference between RTA* and LRTA* is that RTA* stores the second lowest f = g + h value generated by the minimum search. This works well when the hash table is not reused for subsequent

searches toward the same goal. However, if it was reused, the heuristic values stored in the hash table might become non-admissible. LRTA* thus stores the lowest f value generated by the minimum search to enable the hash table to be reused for solving additional problems.

5.8 Summary

In this chapter, all possible collision types have been defined for multi-agent systems where one deadloop condition and five collision types will be considered through the whole thesis. The proposed extension to A* is capable of avoiding not just other moving agents but also static obstacles. Also, the proposed extension allows agents to replan their routes as optimal as possible. Simulations of the algorithm are conducted in different state-space configurations. The algorithm is tested on two minibots in real world, small-scale environments containing obstacles. However, while Super A* has shown how to ensure that two agents do not enter a tunnel together from opposite ends in a static environment, there still remains the question of how to handle the situation where two agents find themselves in a tunnel, heading towards each other. This is due to dynamically changing environments or miscalculated priorities. Therefore, research question 1 has only been partially answered for environments with priority. In the next chapter, we will show that centralized decoupled algorithm solves the deadlock issue in tunnel-like environments and later addresses the second research question identified in the Research Methodology chapter.

Chapter 6

P* and SKP Algorithm

The aim of this chapter is to address research question 2 (Research Methodology chapter): Can we solve deadlock situation in a cooperative way in a tunnel-like environment? This research question follows naturally from the first research question. The first research question led to the identification of all possible collision types and the Super A for avoiding collisions is classified into one of these types. But the question was left open in the previous chapter as to how to deal with deadlock, which falls outside the scope of Super A*. The solutions to the problem of these collision types depend on there being a certain amount of space around the agents that allows collisions to be avoided.*

While we used standard A as a basis for developing a novel method, for handling collision types which is the Super A*, we have seen that there is actually very little in the literature on how to handle collisions in a dynamic environment using A* without recalculating all paths from scratch. In a dynamic, distributed and decentralised scenario, such recalculation is neither desirable nor possible without compromising the goal of autonomous agents. The work presented in this chapter is novel due to its formulation of two algorithms that are designed with autonomous agents in a dynamic*

environment where a collision is detected. Currently, the only classical approach that presents itself is for the central coordinator to plan all agents' moves ahead of any first move, identify possible collisions and then re-plan agents' moves in such a way that collision-free routes are 'guaranteed' once the agents start moving. This is a severe limitation on autonomous, distributed agents. The aim of this chapter is to show how we can allow for dynamic re-planning of routes in real time (while agents are moving) with Super A and other novel methods.*

The previous chapter investigates all possible collision types in multi-agent systems and presents one centralized coupled algorithm solution based on these collision types. This chapter introduces the deadlock issue and proposes two novel real time collision avoidance algorithms, i.e., P and SKP algorithms, for dynamic and decoupled systems. Section 1 introduces the importance of a decoupled collision avoidance approach. Section 2 reviews and describes the limitations of existing algorithms. Section 3 presents the P* algorithm that solves the problem of these collision types with dynamic priority allocation. The SKP algorithm that solves deadlock situations in tunnel-like environments is described in Section 4. The evaluation of P* and SKP is presented in Section 5. Finally, the summary of the proposed algorithms is given in Section 6.*

6.1 Introduction

Multi-agent systems can be used to achieve tasks beyond the capability of an individual agent, especially in the presence of uncertainties, incomplete information, distributed control, and asynchronous computation [43]. To avoid collisions, a multi-agent system needs to include a collision avoidance mechanism to coordinate agents'

actions / movements and ensure that agents, after planning their moves, do not collide with each other when converting their planned moves into actual motion. Some multi-agent systems may include a number of loosely coupled mobile autonomous agents and work under open dynamic environments. Collision avoidance in these situations is more challenging since agents manage their moves independently and may have only a limited capacity to detect the potential risk of collisions. The problem will be more complex if we want to reduce the effect of collision avoidance on the 'optimally' planned routes of individual agents.

In decoupled approaches, agents plan their routes and make decisions independently. Decoupled collision avoidance is more challenging since each agent plans its movements independently, and can only have a limited capability to detect the risk of collisions. The problem will be even more complex if we want to reduce the effect of collision avoidance on the 'optimally' planned routes of individual agents. Ideally, an optimal decoupled system should allow each of the agents involved in a possible collision to make minimum changes to their planned routes so that, after avoiding collision, they can return to their optimal planned routes. Unfortunately, there is no known algorithm which allows decoupled systems to satisfy the global objective of minimum perturbation to a planned route for all agents. For example, in Figure 6.1, agent R1 (heading south) will collide with agent R2 (heading north), and vice versa, when trying to reach their desired targets (red squares). One possible solution is to move one of the agents out of the way to allow the other to pass. So, for instance, R2 can move to the left and then head north before making a diagonal move (R2 north-east) back to the optimal path to reach its target. Another possible solution is that both R1 and R2 make diagonal moves (R1 south-west and R2 north-east) before

moving back to their optimal paths with another diagonal move each after passing each other. If the path cost of a horizontal (or vertical) move and a diagonal move are 10 and 14, respectively, the total path cost of the former solution (74) with two robots is less than the latter solution (76). However, it is not clear how generalizable the former solution is, especially when more than two agents are involved or if there is no space for an agent to move to one side.

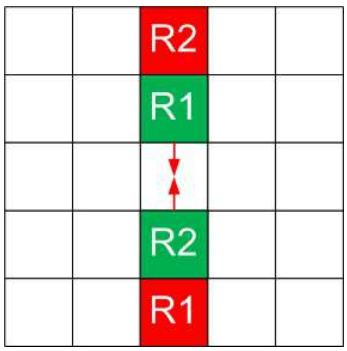


Figure 6.1: A problematic situation for centralized and decoupled approaches where shortest path length is the global objective. The green square is the initial position, the red square is the goal position. R1 and R2 are agent 1 and agent 2.

6.2 Review of Existing Approaches

To overcome some limitations of existing methods, in this chapter we introduce the P* algorithm which can enable individual agents to manage their own movement and achieve peer-to-peer coordination. The P* algorithm (to be described below) differs from prioritized A* in that prioritized A* is run once to find possible collisions, and collisions are resolved before agents move. In the P* algorithm, optimal paths are initially formed using standard A* (the decoupled phase) without considering other agents. Then a peer-to-peer coordination method is used to check possible collisions

one step ahead of each movement without a global view of the entire environment. A dynamic priority queue is determined by a distance remaining rule: the less distance remaining for an agent, the higher priority that agent has. An improved A* algorithm, called Super A*, was developed to resolve any collisions by dynamic priority queuing before the agents move, if possible collisions are detected. This 'plan-evaluate-move' process is repeated iteratively until all agents reach their goals. In other words, the P* algorithm takes into account that the world can change as agents move, so that two agent paths that were previously not involved in collision can become involved in collision if the environment changes as the agents move (e.g. an agent breaks down, or a new agent is placed in the environment). The 'plan-evaluate-move' phases of P* allow dynamic aspects of path finding to be included in prioritized A* planning.

6.3 P* with Dynamic Priority Scheme

In this chapter we propose a novel decoupled collision avoidance algorithm, called P*. This algorithm can facilitate decoupled agents to achieve collision avoidance through the following three steps.

1. Each agent computes its optimal path by using a classical path finding algorithm, i.e., the A* algorithm.

2. Then, agents detect potential collisions based on their limited local views, and use a peer-to-peer coordination approach to make collision avoidance decisions. Also, agents use a path re-planning algorithm, called Super A*, to avoid collisions.

3. Each agent recalculates the current environment using standard A*; if the goal

is achieved, then the algorithm is stopped for that agent. Otherwise, the agent goes back to step (1) for the next step until all agents achieve their goals.

The Super A* algorithm allows agents to avoid dynamic obstacles. However, it does not allow agents to have interactions, and agents have to coordinate their movements according to fixed / pre-defined priorities. This feature cannot properly be adapted to dynamic environments. To compensate for this limitation, P* is proposed here to coordinate agents' movements. The P* algorithm assumes that two agents can share some simple information, which includes their current position, previous position, intention of the next move, and estimated distance remaining. The main idea is to obtain a dynamic priority based on a set of rules. In this work, we consider a decoupled and prioritized path planning approach which plans the paths in the configuration space. First, each agent computes the path with standard A* and without considering the paths of other agents. Then each agent broadcasts its information including the current position, previous position, intention and estimated distance remaining in order to share and coordinate for the deadloop and the possible collision checks. Conflicts between agents are resolved by introducing the Super A* algorithm if P* detects the potential collisions. Otherwise each agent makes its move along the original planned path. A fixed priority scheme determines the order in which the paths for the agents are re-planned. The path of an agent is then planned in its configuration space. The traditional A* search algorithm is then adopted to find the optimal path for each individual agent in the current configuration space (planning). When 'deadloop' and other collision conditions are met, the priority method is used to resolve conflicts (evaluation) before agents are allowed to move (moving) (noted as Video S2). Table 6.1 and Algorithm 2 describe the notations and the pseudo code

of the peer-to-peer coordination algorithm used in P*, respectively.

Table 6.1: P* Notations

Notation	Descriptions
L	Label of agent
C	Agent current position
M	Agent previous position
I	Agent intention
D	Agent estimated distance remaining to the goal
P	Agent priority

Algorithm 2 P* Coordination Algorithm

Input: Input L^K, C^K, M^K, I^K, D^K, $K \in \Re^d, i, j \in K$ for $i, j = 1, 2, i \neq j$
Output: Output **true** or **false**, and P^K

flagged $\leftarrow$ **false**
flagged $\leftarrow$ Call P* - Deadloop Check Algorithm
flagged $\leftarrow$ Call P* - Head-On Check Algorithm
flagged $\leftarrow$ Call P* - Front Sideswipe Check Algorithm
flagged $\leftarrow$ Call P* - Rear Sideswipe Check Algorithm
flagged $\leftarrow$ Call P* - Front-End Swipe Check Algorithm
flagged $\leftarrow$ Call P* - Front-End Sideswipe Check Algorithm
if flagged = **true then**
  if $D^i > D^j$ **then**
    $P[0] \leftarrow L^j$, $P[1] \leftarrow L^i$
  else
    $P[0] \leftarrow L^i$, $P[1] \leftarrow L^j$
  end if
end if
Return flagged

Multi-agent coordination allows a multiple agent system to detect and avoid collisions through a centralized coordination, resulting in a dynamic allocation of priority to help agents re-plan their moves. The dynamic priority scheme is based on various constraints and rules, such as distance remaining and locations previously visited. Compared with the existing algorithms [120, 102], the coordination algorithm can

perform well in open dynamic environments, especially when agents have dynamic goals and dynamically allocated priorities.

6.4 SKP Algorithm

One issue with the previous algorithms is that they are not capable of dealing with the deadlock situation shown in Figure 6.2. If two agents move towards each other in a narrow passageway, the lack of space in the tunnel will prevent one agent from moving out of the way. Ideally, agents should temporarily cooperate, leading to one agent re-planning a way to escape this deadlock while the other agent can continue to move to its target location. The reason for deadlock is that previous algorithms must calculate a complete route to the target location. In the calculation, the expanded node is checked with the nodes in the CLOSE list. If there is a matched node, this means the agent comes from that node (its parent) and the algorithm will remove the parent node to ensure that the agent does not return to where it has just come from. Instead, other solutions are searched for, but in Figure 6.2 there is no alternative route, hence the deadlock.

A simple solution to this is to search only one step lookahead without fully computing a route to the destination. In other words, when two agents encounter the deadlock, one agent follows an incomplete search strategy, which adds adjacent unoccupied nodes into the OPEN list, then selects the node with the lowest cost and executes the move in order to escape the deadlock. This approach can deal with the deadlock in a narrow passageway like a tunnel. That is, if R1 performs an incomplete search and R2 performs a complete search (because R2 has higher priority than R1), R1 will move one step backwards to allow R2 to move forward. This is repeated

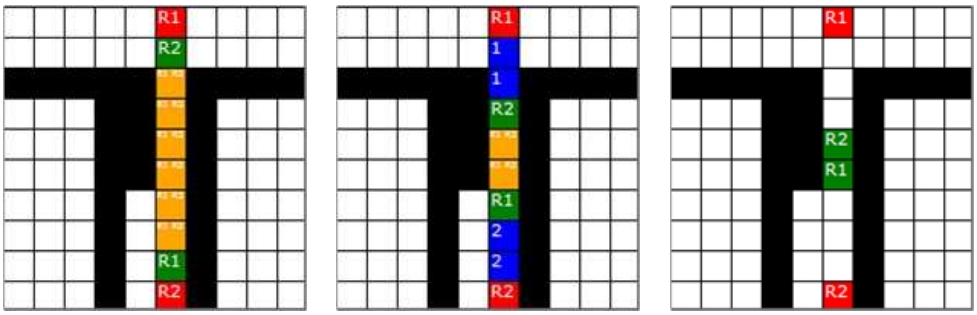


Figure 6.2: Illustration of deadlock when two agents move towards each other in a tunnel. R1 and R2 denote two agents, and G1 and G2 their destination nodes, respectively. (a) shows the initial situation of two agents. (b) shows the two agents following their optimal paths as signified by the numbers in the squares. (c) shows the two agents detecting the collision one step ahead and stopping due to no solution being found.

for as many steps as it takes to resolve the deadlock, at which point R2 calculates a complete path again. Pseudocode for the SKP algorithm is shown in Algorithm 3.

The strategy repeats a 'plan-evaluate-move' process to plan agents' routes and avoid potential collisions. In detail, collision avoidance is achieved through the following steps:

1. Each agent computes its optimal path by using a classical path finding algorithm, i.e., the A* algorithm.

2. Each agent reports to the coordinator its current node position, previous node position, intended next node position and estimated remaining distance to the goal node.

3. The coordinator detects potential deadloop and collisions based on agents' intentions (i.e., the nodes the agents want to move to). If no collision or deadloop is detected, agents go to Step (6), otherwise they go to the next step.

Algorithm 3 SKP Algorithm

```
1: N_open ← n_0, N_close ← ∅
2: loop
3:    pn ← Compute the lowest cost of node, in N_open
4:    if pn = n then
5:       N_close ← N_close ∪ {pn}
6:       Return N_close
7:    else
8:       for all neighbours n_new do
9:          if n_new ∈ N_open and G_new < current G then
10:            G ← G_new
11:            Continue Loop
12:         end if
13:         if n_new ∈ N_close and G_new < current G then
14:            G ← G_new
15:            Continue Loop
16:         end if
17:         N_open ← N_open ∪ n_new
18:      end for
19:   end if
20:   N_close ← N_close ∪ {pn}
21:   if N_open > 0 then
22:      N_close ← Popup the node, in N_open
23:      Return N_close
24:   end if
25: end loop
```

4. Agents with potential collisions or deadloop will use the Super A* algorithm, and with a potential deadlock will use the SKP algorithm, which is described in Algorithm 3, to re-plan their routes in a decoupled manner. Agents with less remaining distance will get higher priority for path planning purposes.

5. Agents will then report their re-planned intentions to the coordinator. Steps (3)-(5) are repeated until no collision or deadloop can be detected by the coordinator.

6. Each agent moves to their intended node. If the goal node is achieved, then the algorithm is stopped for that agent. Otherwise, the agent goes back to Step (1)

and repeats Step (1)-(6) until all agents achieve their goals.

6.5 Evaluation for P* Algorithm

Our approach has been tested on real robots and in extensive simulation runs to answer the following questions. (1) Practicability: is our approach relevant and applicable to real robot systems? (2) Solvability: does our approach succeed in finding valid collision-free multi-robot paths? (3) Optimality: is our approach able to generate the best paths? All experiments were carried out using different configuration environments. The proposed method is applied to a two robot system with the deadloop and collision conditions described above. The control architecture is through networked robotics to allow our mobile robots to communicate with each other via WiFi.

6.5.1 Experiment 1: P* in Simple Collision Avoidance

In Experiment 1, we implemented the scenario in Figure 6.1 in a real robot environment. The goal of the experiment is to demonstrate the applicability of our approach to real robot systems. This experiment was carried out using the Rovio robots of WowWee Technologies. The task of the robots was to find the optimal / shortest path and move from their initial positions to their goal positions without collision. Figure 5.5 shows one experimental robot and one experimental physical space, a 5 by 5 space with each grid of 50cm by 50cm. The video link is http://youtu.be/w24SkiYuh7g.

In Figure 6.3(a), the robots are deployed on two sides of the field and have to

move to their goal positions on the other side using P*, avoiding dead looping (Figure 5.2) and collisions. Figure 6.3(b) shows one robot moving away from its optimal direction in the initial situation. As can be seen in Figure 6.3(c), the path of one robot is changed in order to avoid collisions with each other. For the optimal paths computation, one robot has to take a detour around the other and then return to its optimal path as quickly as possible (see Figure 6.3(d)). At the end, the two robots reached their goal positions (see Figure 6.3(e)). This is the optimal resolution to this possible collision without introducing a sideways move or a front-end swipe or sideswipe.

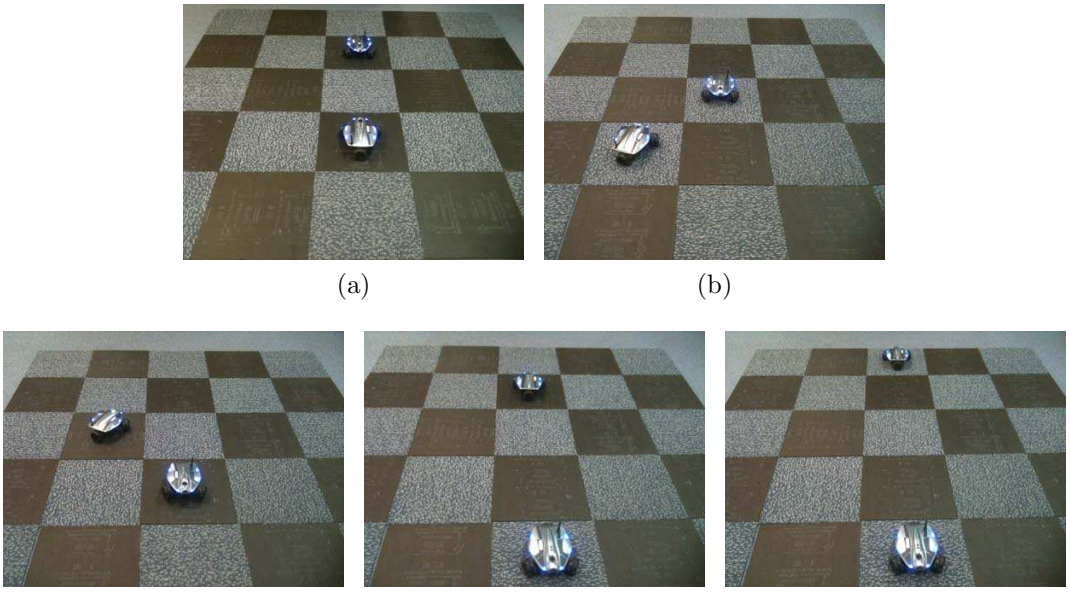


Figure 6.3: An application example with the Rovios of the WowWee Technologies: (a) shows the initial situation of two robots. (b) shows one robot moving left to avoid collision. (c) depicts the two robots passing each other with no collision. (d) shows one robot returning to its optimal path and the other robot at its goal position, and (e) shows the two robots in their goal positions.

6.5.2 Experiment 2: Comparison of P* and Super A*

In Experiment 2, we compared the performances of the P* algorithm and the Super A* algorithm. Two scenarios, i.e., with collision and without collision, were simulated in a 10 by 10 space.

Scenario 1: With Collision

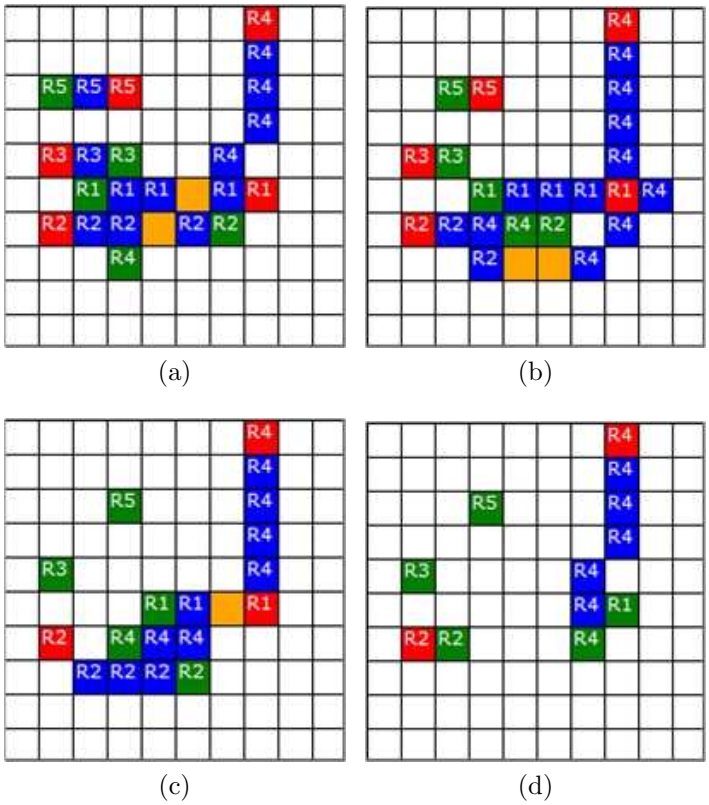


Figure 6.4: Illustration of Super A* for the fixed priority with collision conditions for 5 agents: (a) shows the original calculated paths. (b) shows the suboptimal collision avoidance planning with fixed priority between R2 and R4; R2 has higher priority than R4. (c) and (d) show Super A* (with fixed priority) and optimal collision avoidance.

In the first scenario, we simulated five agents and randomly selected an initial

and goal position for each agent. In the experiment, both Super A* and P* allow agents to cross-traverse each other's paths without collision. Figure 6.4 shows the path re-planning result of Super A*. In Figure 6.4(a), the blue squares represent the calculated paths of the agents, and the orange squares the potential collision risk where the paths of some agents overlap. Figure 6.4(c) and (d) show that agent R1, agent R3 and agent R5 can still take the optimal paths, but agent R2 and agent R4 take a sub-optimal path to avoid collisions.

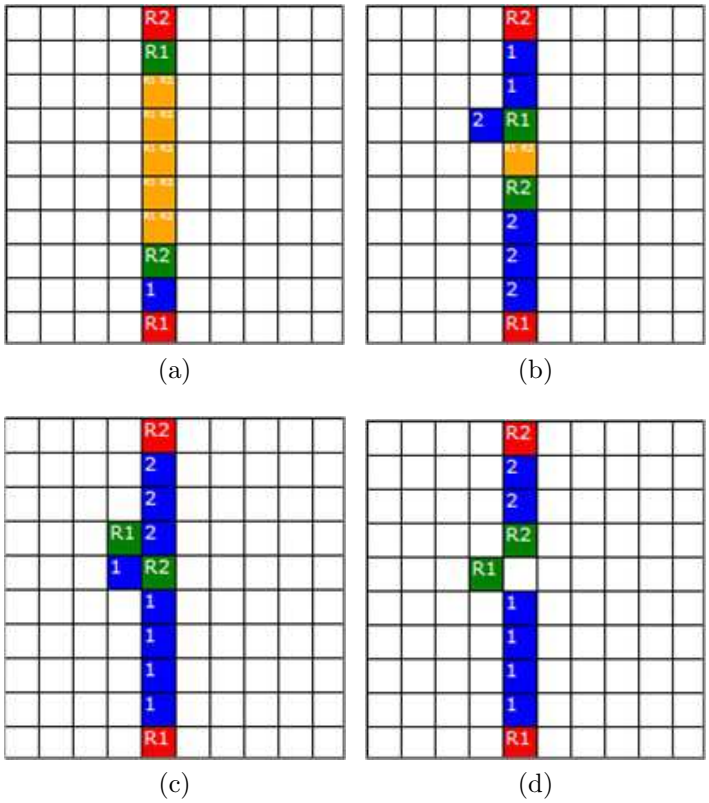


Figure 6.5: Illustration of P* for the dynamic priority with collision condition for 2 agents: (a) shows the original calculated paths using standard A*. (b) shows the suboptimal collision avoidance re-planning with dynamic priority (rule-based, less distance remaining higher priority), here, R2 has a higher priority than R1. (c) and (d) show P* (with dynamic priority) and optimal collision avoidance.

The path re-planning results of the P* algorithm are shown in Figure 6.5 for two agents involved in a potential deadloop situation. It can be seen that R1 lets R2 take the optimal path, and R1 selects an avoidance strategy that allows it to return to the optimal path after collision is avoided. In addition, the P* algorithm can also cater for a combination of possible collisions. For instance, as shown in Figure 6.6, if a change in the goal states of two agents leads to a deadloop condition, the P* algorithm can resolve the problem effectively.

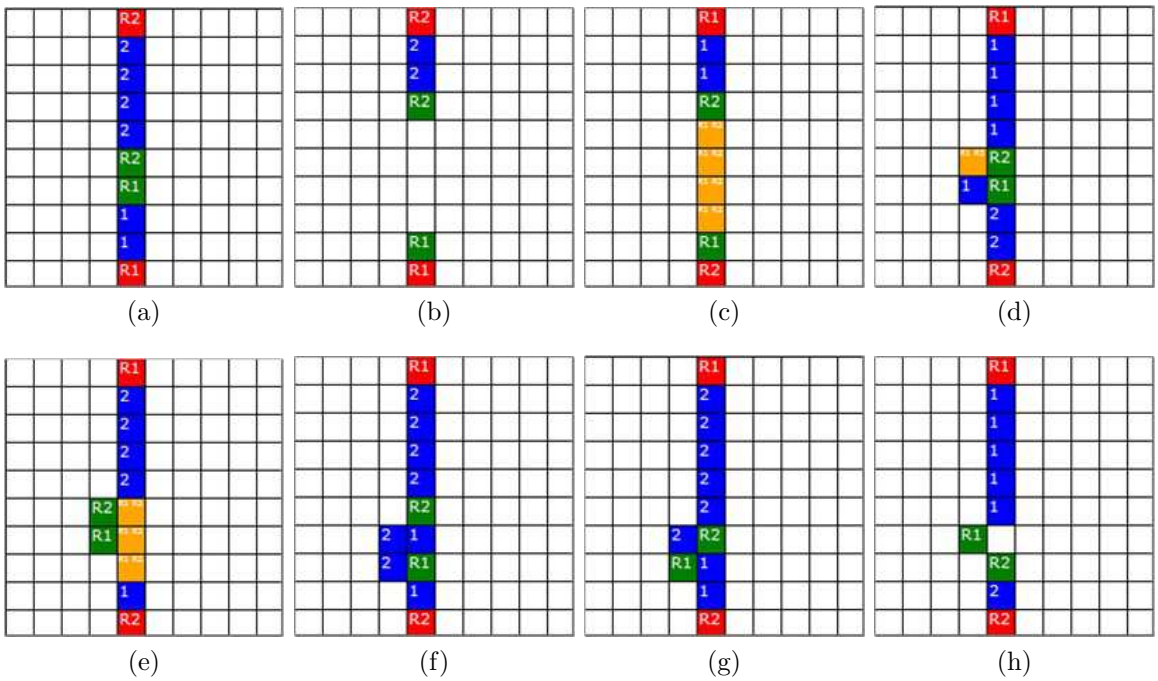


Figure 6.6: Illustration of P* for the dynamic priority with deadloop and other collision conditions for 2 agents: (a) shows the original calculated paths using standard A*. (b) shows the agents following the established optimal paths to move and suddenly their goals are changed with swiping each other as seen in (c), robots recalculate their paths. (d) and (e) show two agents encountering the deadloop condition. (f) shows two agents avoiding deadloop condition using P*. (g) shows suboptimal collision avoidance with dynamic priority (less remaining distance, so higher priority, R1 moves away from the optimal path to give way to R2). (h) shows P* (with dynamic priority and optimal collision avoidance).

Scenario 2: Without Collision

The second scenario in Experiment 2 was designed to demonstrate that the P* algorithm is flexible enough to achieve path re-planning in more complicated environments. Figure 6.7(a) demonstrates the environment of two agents needing to pass through a tunnel to reach their goals. In Figure 6.7(b), R1 gets to the tunnel first, so R1 gets the priority to go through the tunnel. On the other hand, R2 follows without stopping and waiting, as shown in Figure 6.7(c). This is an example of allocating priority based on time. That is, P* not only considers the optimal path cost but also takes the optimal time cost into account. Finally, the agents find clear paths to achieve their goals, as shown in Figure 6.7(d).

6.6 Evaluation for SKP Algorithm

In this evaluation, deadlock in a tunnel environment is simulated. The scenario demonstrates the performances of SKP algorithm, see Figure 6.8.

6.7 Discussion

In a multi-agent system, agents use path finding algorithms to find the shortest / optimal paths from a start position to a goal position. In the context of multiple agent path planning, there is a lack of understanding of whether collision avoidance and re-planning should be included at a global level or whether collision avoidance should be left to agents to negotiate at the local level. The proposed method described in this chapter is capable of detecting the deadloop and other collision types before allowing agents to move, and calculates priority based on constraints adapted to the dynamic

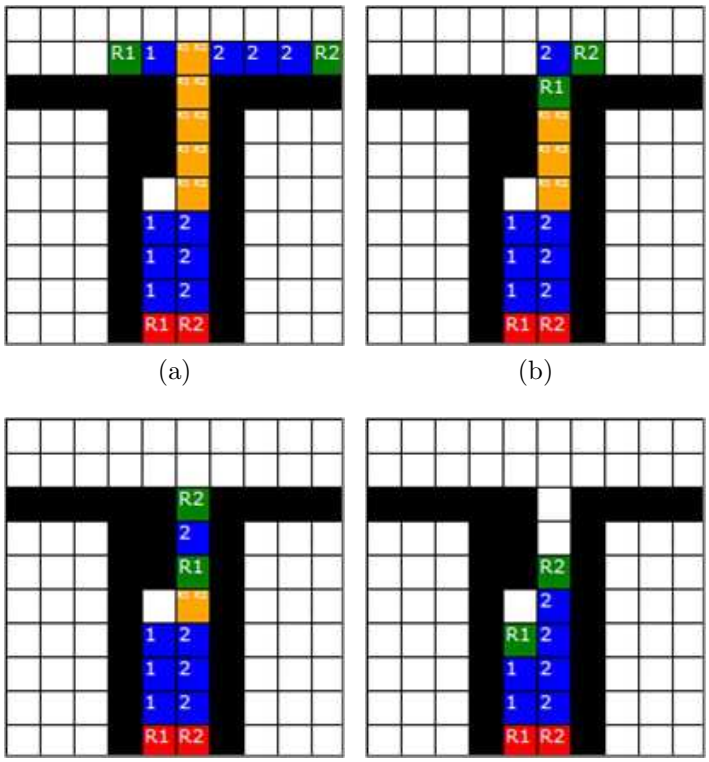


Figure 6.7: Illustration of priority without collision condition. R1 reaches the tunnel first and therefore receives higher priority. R2 follows (b) and (c) without collision, until a clear path is found to achieve their goal (d).

environment. In other words, the P* algorithm integrates standard A*, peer-to-peer coordination and Super A* to resolve the conflicts for re-planning paths. The Super A* algorithm assumes that re-planning should be built into A*, taking into account the many different types of collision possible, so that global requirements of optimality can be addressed. Due to major increases in computer speed since the early days of robotic planning, the major bottleneck now can be expected to lie in inter-robotic communications. Leaving agents to deal with collision avoidance locally is subject to the 'horizon effect', when a new danger presents itself that could lead to robots

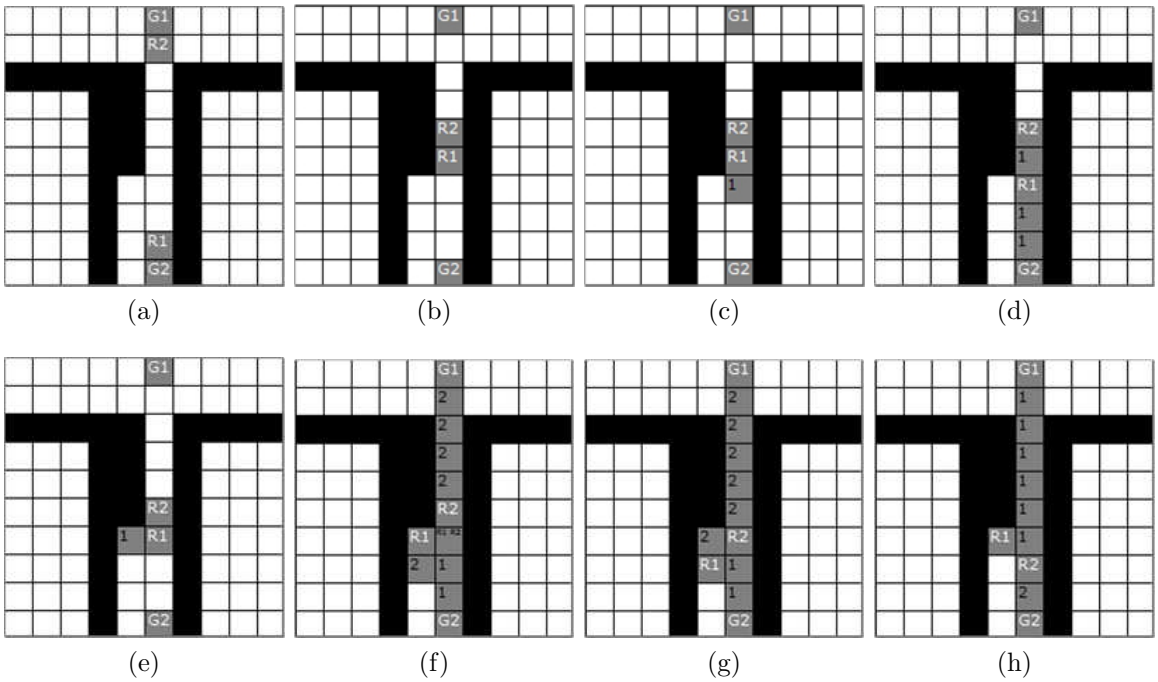


Figure 6.8: Illustration of SKP* for the tunnel deadlock situation for 2 agents: (a) shows the initial situation of two agents. (b) shows the two agents following their optimal paths using standard A* to move. (c) R1 moves out from the path of R2 one step. (d) R2 moves forward to the goal, and then the deadlock is formed again, R1 continues to move out in (e). (f) and (g) R1 moves out of the way and R2's path is clear. (h) shows the agents completing the temporary cooperation.

taking global non-optimal re-planned routes, which could further increase the risk of collisions. The proposed algorithm has been shown capable of dealing with both static and dynamic obstacles. Each agent is able to find the optimal path or, if it has to move off the path to avoid collision, to resume its optimal path. Our extensions to standard A* involve the use of a step-by-step recalculation once the full paths are initially calculated. Agents are allowed to progress on their optimal paths until collision is detected one step ahead or an obstacle is encountered. This delay in collision detection allows the system to be in an optimal state for as long possible and to take into account obstacles that appear from over the horizon. Once collision is detected,

the appropriate avoidance behavior is taken and the agents return to their optimal paths as quickly as possible. The extensions to A* rely on a networked robotics architecture that allows collision avoidance to be built explicitly into the re-planning process.

6.8 Summary

The algorithm P* allows a multiple agent system to detect and avoid collisions through peer-to-peer coordination, resulting in a dynamic allocation of priority to help agents re-plan their moves. We also present simulations of the proposed peer-to-peer coordination to demonstrate the effectiveness of the P* algorithm under a number of different collision conditions. The proposed dynamic priority scheme is based on various constraints and rules, such as distance remaining and locations previously visited. Also, the P* algorithm is tested on two minibots in real world, small-scale environments containing obstacles. The results show that the collision avoidance and re-planning approach is useful for managing possible collisions between agents, or other groups of autonomous agents, working independently in a shared physical environment and needing to traverse the environment to undertake and complete their tasks.

To conclude this part of the decoupled collision avoidance approach, we have shown that, while all possible collision types are identified and resolved by Super A* in the previous chapter, the deadlock issue was not considered by the coupled approach. Also, the coupled approach is a fixed priority scheme for avoiding collisions. To contrast with the Super A* coupled approach, the solution to the deadlock depends on there being a dynamic priority scheme that allows agents communicative

and cooperative methods for avoiding a deadlock situation. The previous and current chapters represent two types of centralized approach to solve collision issues (including deadlock) based on our defined collision types, coupled and decoupled. In the next chapter, we will attempt to apply the presented algorithms for a multi-agent architecture.

Chapter 7

Swarm Robotic Group Formation

The previous two chapters focus on centralized collision avoidance. The aim of this chapter is to address the third research question: how can paths be planned for a moving goal in swarm robotic group formations? Also, there has been relatively little nature inspiration in this thesis so far. The materials covered in the previous two chapters are essentially major and provide novel extensions to existing search algorithms for single agents. Now that we move to swarm robotics, the thesis also makes its first move towards introducing inspiration from nature to help solve the collision avoidance problem.

This chapter gives a further evaluation on swarm robotic formation based on the proposed algorithms, such as Super A, P*, SKP and the coordination strategy. Section 1 introduces swarm robotics. Section 2 describes the background. In Section 3 a coordinator-based (centralized) strategy is presented. Further evaluations are demonstrated in Section 4. A performance discussion is given in Section 5.*

7.1 Introduction

Swarm robotics is a new approach to the coordination of multi-agent systems and consist of large numbers of relatively simple robots that are capable of performing a task cooperatively to achieve a common goal [150]. The idea is biologically inspired from nature. For example, ants by themselves may seem to act randomly and without any discernible purpose. However when the collective interactions among ants are taken together, a collective intelligence and behaviour that have the capacity for solving many problems will emerge, such as searching for food and building a nest. Swarm robot systems can be used to achieve tasks beyond the capability of an individual robot, especially in the presence of uncertainties, incomplete information, distributed control and asynchronous computation [56]. In many swarm robot systems, forming and maintaining a desired formation can result in significant benefits, such as a school of fish reducing the chances of an individual member straying too close to a predator or a pack of wolves cooperatively hunting prey.

To avoid collisions in swarm robot systems, path planning needs to include a collision avoidance mechanism to coordinate robots' actions / movements and ensure that robots moving plans do not collide with each other when converting into actual motion. In difficult environments with one passageway and multiple robots, an ideal multi-robot system not only requires collision avoidance, but also cooperative behaviours. In this work, we propose a coordinator based strategy to plan and control agents' movements in the context of these collision types with the goal of demonstrating how robots can form a particular pattern once they reach the goal node. The strategy repeats a 'plan-evaluate-move' process to plan agents' routes and avoid potential collisions due to same cell occupancy and possible sideswipes while

still finding optimal paths from the starting position to the destination position for each robot within a formation.

7.2 Background

Silver [120] describes an algorithm, i.e. WHCA*, which builds a more accurate heuristic determination for the multi-agent planning problem by solving the single-agent problem. To further reduce computational costs, it performs a windowed search through the cooperative space, reverting to a single agent computation after the window is exceeded. This idea chooses an ordering of agents, and plans a path for each agent that avoids conflicts with previously computed paths by checking against the 'reservation table'. The reservation table is a data structure that signifies, for each cell of the space-time map, whether that cell is available or reserved. However, in addition to the extra cost required to maintain this data structure for large spaces containing a large number of agents, a reservation table is not feasible in cooperative pathfinding where the map is dynamic (obstacles appear and disappear) and the goal state can change at any time. Also, the window size is not easy to handle, especially in a dynamic environment. If the window size is small, this may lead to the bottleneck situations not being solved, and if it is large, some redundant searches may be performed.

In this chapter, the proposed algorithms are applied to each agent independently, which consider the moves of all agents simultaneously in a time step with each state potentially has 9^n legal actions. Each of these legal actions is a solution to the constraint satisfaction problem in which each agent must be assigned a move from $\{E, S, W, N, NE, SE, SW, NW,$ and $wait\}$, and there are constraints between sets of

legal moves. For example, these legal moves must not lead to two agents colliding. With respect to preventing collisions, the algorithms presented in this chapter are intended to cope with collisions with one step lookahead in various situations, such as tunnel-like environments.

7.3 Collision Avoidance Strategy

In this chapter, we propose a coordinator-based (centralized) strategy to prevent collision. The strategy repeats a 'plan-evaluate-move' process to plan robots' routes and avoid potential collisions. Collision avoidance is achieved through the following steps: 'distributed' signifies that each robot performs this step independently of any other robot, 'centralised' means that the central coordinator is involved in calculating the step, 'coupled' means that a single planning system is used, and 'decoupled' means that each robot plans its own path.

1. Each robot independently computes its optimal path by using a classical path finding algorithm, i.e., the A* algorithm (distributed, decoupled).

2. Each robot reports to the coordinator its current node position, previous node position, intended next node position and estimated distance remaining (EDR) to the goal node (distributed, decoupled).

3. The coordinator detects potential deadloop and collisions based on robots' intentions (i.e., the node each robot wants to move to). If no collision or deadloop is detected, it goes to Step (6), otherwise it goes to the next step (centralized, coupled).

4. Robots with potential collisions or deadloop will use the Super A* algorithm, and with deadlock will use SKP algorithm, which are described above, to re-plan their routes in a decoupled manner (centralized, decoupled). Robots with less remaining distance will get higher priority for path planning purposes (centralized, coupled).

5. Robots will then report their re-planned intentions to the coordinator (distributed, decoupled). Then they repeat Steps (3)-(5) until no collision or deadloop can be detected by the coordinator.

6. Each robot moves to its intended node. If the goal node is achieved, then the algorithm is stopped for that robot (distributed, decoupled). Otherwise, it goes back to Step (1) and repeats Steps (1)-(6) until all robots reach their goals.

7.3.1 Multi-Robot Coordination

The coordinator-based approach is a decoupled approach to plan the paths individually for each agent and uses centralized communication to coordinate robots' movements on demand, such as when collisions are encountered. The coordinator-based approach assumes that two robots can share some simple information, which includes their current position, previous position, intention of next move and estimated distance remaining. The main idea is to obtain a dynamic priority based on a set of constraints. First, each agent computes the path with standard A* algorithm and without considering the paths of other agents. Then each robot broadcasts its information including the current position, previous position, intention and estimated distance remaining to share and coordinate for the deadloop and the possible collision

checks. If two or more robots are involved in a potential collision, conflicts between them are resolved through Super A* algorithm. Otherwise, each agent makes its move along the original planned path. The 'plan-evaluate-move' strategy for reaching a goal has the added advantage that it can deal with dynamic situations, such as the goal being moved or an obstacle appearing suddenly. Coordination notations and Pseudocode are shown in Table 7.1 and Algorithm 4, respectively.

Table 7.1: Coordination Notations

Notation	Descriptions
DR	Distance remaining from the current position to the goal position
$Priority$	Priority queue of agents starting with '0' index, it is the first priority
$Label$	Agent label

Algorithm 4 Multi-Robot Coordination

Input: $Label$, $Agent'sPath$
Output: $Priority$

$isCollided \leftarrow$ **false**
$isCollided \leftarrow CollisionCheck()$
if $isCollided =$ **true then**
    if $DR^i > DR^j$ **then**
        $Priority[0] \leftarrow Label^j$, $Priority[1] \leftarrow Label^i$
    else
        $Priority[0] \leftarrow Label^i$, $Priority[1] \leftarrow Label^j$
    end if
end if
Return $isCollided$

To summarize, multi-robot coordination allows a multiple robot system to detect and avoid collisions through a centralized coordination, resulting in a dynamic allocation of priority to help robots re-plan their moves. The dynamic priority scheme is based on various constraints and rules, such as distance remaining and locations previously visited. Compared with existing algorithms [102, 120], the coordination algorithm can perform well in open dynamic environments, especially when robots

have dynamic goals and dynamically allocated priorities.

7.3.2 Swarm Super A*

Unlike the traditional A*, Swarm Super A* involves another heuristic cost which is calculated from agent start node to the current expanded node. The logic is same as Super A* only the function cost formula is changed. By doing so, the cost function is $F = G + H_{c2g} + H_{s2c}$, with G is the gone cost, H_{c2g} is the heuristic cost from the current expanded node to the goal node and H_{s2c} is the heuristic cost from the start node to the current expanded node. This leads to boosting performance of finding the optimal path and reduces the chance of cyclic searching. Furthermore, Swarm Super A* enables robots to achieve a desired formation when the goal is reached while simultaneously avoiding fixed and dynamic obstacles, including themselves. Unlike other behaviour-based formation approaches [9, 85], communication among robots is not required and each robot does not have a specific position to maintain formation, i.e. the proposed formation does not have strict requirements on fixed positions for individual robots. The desired goal formation in this chapter is occupying and fully encircling the goal node, resulting in a 3 by 3 formation of robots (9 robots). Also, Swarm Super A* is capable of dealing with groups of robots needing to adopt the desired goal formation at different goals simultaneously. The idea is motivated by swarming behaviour in the real world, e.g. ants circling a food source.

7.4 Simulations

In this section, we show the results of two types of simulations. In the first simulation, we evaluate the performance of goal formation control which simulates swarm robots starting at different random positions encircling a common destination node to form a 3 by 3 shape. The second simulation is swarm robots passing through a one-passageway tunnel-like environment and then managing to keep 3 by 3 formation. Collision avoidance is also taken into account in swarm robots. Goal swarm formation around the goal only takes place when each robot reaches the goal node. This is in contrast to group swarm formation where the robots move in formation to a desired goal state (e.g. [30]).

7.4.1 Simulation 1: Formation with Collision Avoidance

In Simulation 1, we implement the scenario of swarm robots forming a 3 by 3 shape at the goal node. 9 robots are randomly located on the grid. The task of the robots is to find the optimal / shortest path and move from their initial positions to the common goal and form a 3 by 3 shape around the goal node without collision. The video link is http://youtu.be/oYMNyPIGtxw. Note that the algorithms can cope with the goal being changed while the robots are still in motion.

7.4.2 Simulation 2: Tunnel Environment

In Simulation 2, two scenarios are simulated. The first scenario demonstrates the performance of the algorithms when the goal is changed in real time, leading to re-planning in a 6 by 6 space. The algorithms can cater for a combination of possible

collisions. For instance, as shown in Figure 7.1, if a change occurs in the goal state of any robot, the algorithms can resolve the problem effectively. The video link is http://youtu.be/89EoYBgdTl8.

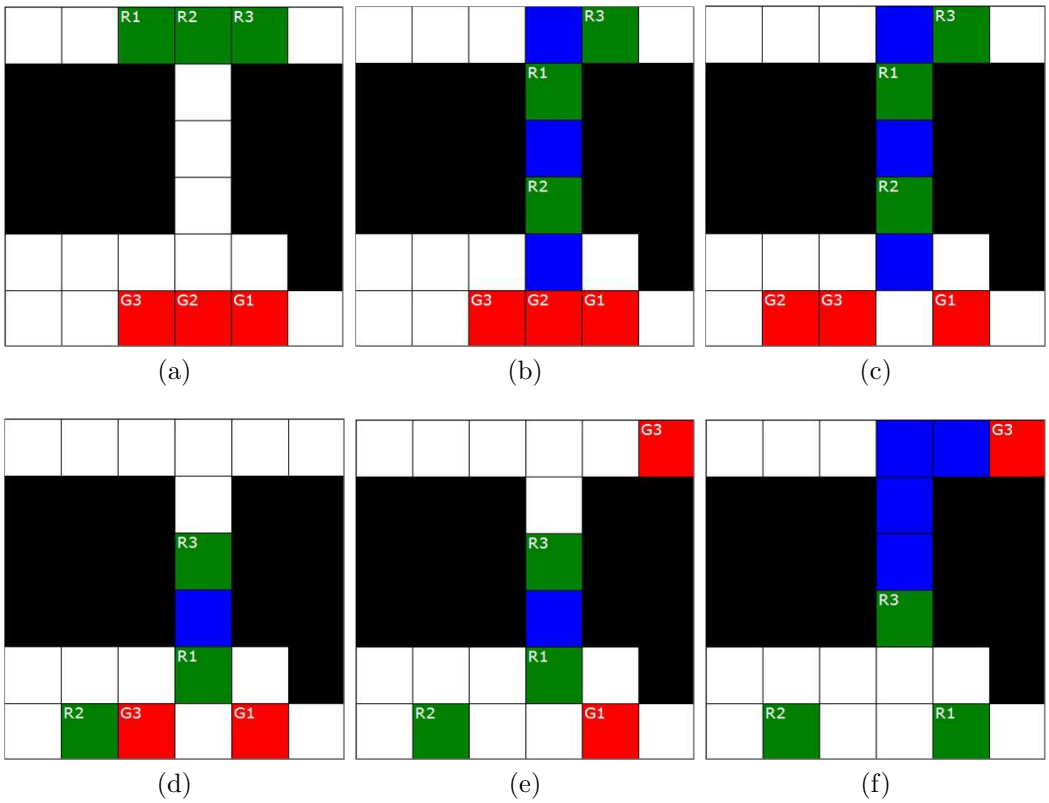


Figure 7.1: Illustration of real-time goal changing in a tunnel-like environment for a three-robot situation. (a) shows the initial positions. (b) shows the robots following their established optimal paths. (c) shows R2's goal changed and R2 recalculate its path in (d). (e) shows R3's goal changed and R3 recalculate its path in (f).

In the second scenario, 9 robots are initialized on one side of the tunnel and the common goal is set on the other side of the tunnel, as shown in Figure 7.2. The robots move sequentially through the passageway and achieve the desired formation around the goal node. Robots only pass one at a time through the tunnel in case the agent itself breaks down or the goal is suddenly changed to be on the other side

of the tunnel (in which case, only one robot in the tunnel needs to turn back). The video link is http://youtu.be/-T-VWN5R0CE.

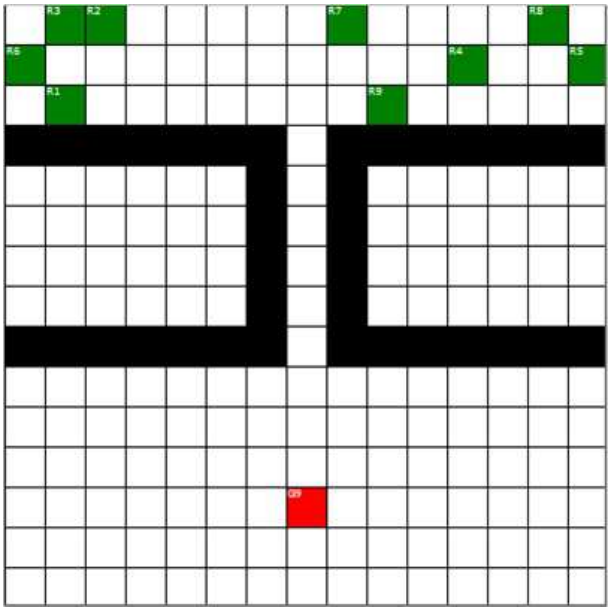


Figure 7.2: The initial positions of 9 robots.

7.5 Discussion

We expanded the configuration space to a 50 by 50 grid involving 99 robots (11 groups), with 10% density of obstacles. All start, goal and obstacle positions were generated randomly. The algorithms were able to move all 99 robots to their desired 11 destination nodes avoiding collisions and obstacles (static and dynamic). Multi-group robot formation simulations are shown in the video link http://youtu.be/xx6sVvs1p-E.

For a 50 by 50 grid and 99 robots with 10% obstacle density, the predicted number of node expansions is 366,796, whereas the actual number of expanded nodes

is 1,220,733. The reason for the discrepancy is that a wait action is not taken into account in the formula. That is, when there is no path to the goal from the robot's current position and given the current set of obstacles, the robot must wait until there is a change in the state of the configuration space and then search again. This process may search duplicate nodes several times.

7.6 Summary

In this chapter, we have further presented swarm behaviour on robotic group formation, which is based on Super A*, SKP algorithms and the proposed coordination strategy. The strategy is tested on a series of challenging, tunnel-like environments. The results show that the strategy is useful for managing possible collisions between agents and allowing agents to escape deadlock in a shared physical and dynamic environment while at the same time adopting a formation once the goal node is reached.

To conclude this part of formation in multi-agent systems, we have shown that, our approach gives one possible solution to the perspective of the nanobots scenario introduced in Chapter 1. Even though the solution has not much inspiration from nature and is limited to autonomous, non-centralized, non-global, non-communicative collision avoidance, we have attempted extending the existing approaches where necessary to make them more suitable for collision avoidance and implement a nanobots scenario. In the previous three chapters of centralized approaches for collision avoidance, we have successfully answered research questions 1, 2 and 3. Following the process of our research methodology, we have moved towards the collision avoidance approach from a centralized to a decentralized strategy. In the next chapter, we will introduce a novel nature-inspired decentralized method for homogeneous collision avoidance.

Chapter 8

A Human-Inspired Collision Avoidance Method

In the previous 3 chapters, centralized methods were studied as necessary for the understanding of dynamic collision avoidance. It is possible that the design is not extendable to a truly intelligent multi-agent system. However, the centralized work on collision avoidance, which we believe is novel, will need to be embedded into any decentralized design. In particular, the work has shown that existing techniques for detecting collisions do not capture all collision types. On the other hand, the limitations of our centralized approaches are lack of autonomy, and the growth of communication overhead by agents. Dealing with these crucial aspects will now form the rest of this thesis and can be expected to lead to a genuinely novel multi-agent architecture. The aim of this chapter is to address the fourth research question: How can a large number of agents autonomously and equally deconflict paths in dynamic and uncertain environments? In the literatures, there has been relatively little on nature inspiration for collision avoidance, especially the methods with no communication, no priority, and local views (evidence shown in Chapter 1). Now that we move to nature inspiration, the thesis also makes its move towards introducing inspiration from nature to help

solve the collision avoidance problem.

In this chapter a novel and dynamic rectangular roundabout ('rectabout') collision avoidance method based on human behaviour is presented for multiple, homogeneous, autonomous and mobile agents. Section 1 briefs the idea of rectabout. Section 2 provides an overview of prior work on collision avoidance. Section 3 provides a brief summary of minimal predicted distance and definition of conflict and collision in our approach. In section 4 we describe the rectabout collision avoidance maneuver for multiple mobile agents. Section 5 presents the experimental (simulation) results of the obtained collision avoidance maneuver. Finally, section 6 gives the summary of this work.

8.1 Introduction

As noted previously, collisions between multiple, autonomous agents is one of the main problems in decentralized, distributed task cooperation. The collision avoidance problem arises when the environment is dynamic and to reach their destination agents need to use paths that conflict with other agents' paths on specific moves. Decentralized collision avoidance in these situations is more challenging than centralized approaches since autonomous agents must manage their moves independently and may have only a limited capability to detect the potential risk of collision. Moreover, another problem is having no communication while avoiding collisions since true autonomy involves no central coordinator and no communication between agents. A secondary problem is how to ensure that agents resume their paths after collision avoidance to still reach their goals effectively and efficiently.

In this chapter, collision detection is inspired by studies on how human pedestrians detect possible collision, and collision avoidance is inspired by the use of roundabouts for resolving potential vehicle collisions at road intersections. When a vehicle approaches a roundabout, drivers adopt a specific set of autonomous but shared procedures for negotiating the roundabout to continue on their routes while avoiding collision with other vehicles already on the roundabout. Drivers do not require a central command and control structure or communication with each other to avoid collision. While roundabouts are concrete in the real traffic world, the temporary roundabouts used here by agents are virtual: they do not actually exist in the configuration space but do exist temporarily on the paths of each agent involved in a possible collision before disappearing from each agent's paths once collision is avoided. No other agent sees this roundabout. Because the configuration space used in my experiments can be regarded as consisting of rectangular grids, the virtual roundabout necessarily becomes straight-lined and rectangular, hence it is called a 'rectabout'. Moreover, real roundabouts can vary in size depending on various parameters such as approach speed, one-lane or multi-lane, number of roads leading to that intersection, etc. In our experiments below, all agents move at the same speed and the size of their local views will determine the size of the roundabout. The larger the local view, the further ahead the agents can plan and therefore the larger the roundabout can be to ensure a smooth deviation from a planned straight line for collision avoidance. However, for the experiments below, the local view is fixed for all agents and therefore the roundabouts are of fixed size.

A desirable roundabout collision avoidance system has to satisfy the following principles. (1) Flexibility in size: since the degree of collision risk depends on the

distance between the two agents, the system should have the ability to shift its scale according to the distance of the nearby agents. As noted above, our experiments below assume fixed size roundabouts for demonstration of feasibility and further work will be required to evaluate flexibility in roundabout size. (2) Orientation: The topology of the roundabout must be such that it is correctly oriented to deal with the conflict. That is, the location and orientation of the roundabout must be able to deal with two agents approaching each other from any angle and not assume a fixed entry and/or fixed exit points. (3) System adaptability: the system can be used for resolving any type of possible collisions for multi-agent systems. (4) System independence: the system should not be restricted to deal with a predetermined number of agents, or a predetermined and fixed configuration space.

In this chapter we use Minimum Enclosing Rectangle (MER) to support these principles. According to Das et al. [23], given a set of points $P = \{p_1, p_2, ..., p_k\}$ with $p_k \in \Re^2$, the minimum enclosing square (or rectangle) of P is the smallest square (or rectangle) that contains all points of P. For the purposes of this work, the smallest square or rectangle is defined to be the smallest rectangle that contains a given number k such that $\frac{n}{2} < k \leq n$ of x-consecutive points in a set of n points in the plane. The problem of computing *k-square* and *k-rectangle* has been investigated since 1991 (for a review, see [1, 31, 115, 23, 79]). MER has been applied in various areas, such as pattern recognition [96], facility location [29], similarity search [88, 24] and collision detection [75]. In order to classify the *k-square* with respect to the number of points η present on its boundary, Das et al. [23] investigated all different possibilities of *k-squares*. As a result, no *k-square* is possible with $\eta = 0$ or 1. The only possibility with $\eta = 2$ is that the two points appear at the two diagonally opposite corners of

the corresponding *k-squares*. In this study, $k = \eta = 2$ is the MER or MES that the agents are searching for.

The proposed method builds on an implicit assumption that other agents make similar collision avoidance reasoning via MER. That is, knowledge of MER is shared by all agents. It consists of two components: Minimal Predicted Distance (MPD) detection and MER rectabout collision avoidance algorithm. The MPD is a metric inspired by real human pedestrian collision avoidance behaviour (for a review, see [92, 93] and more details below). As far as we are aware, this is the first time that MPD has been used for addressing collision problems in multi-agent systems. The MER-based rectabout collision avoidance algorithm is a pairwise approach which computes and re-plans agents' moving direction by following a 'keep left' rule at the rectabout. This rule can be changed if necessary to keep right. The proposed collision avoidance method is demonstrated for multi-agent systems using different types of collision investigated in [76]. So far, the simulations indicate that the proposed approach generates collision-free motions.

We envisage the proposed approach being of use in autonomous cars [145], such as Google's driverless car [125]. If in the future all cars are autonomous (to compress more traffic onto increasingly busy roads), the proposed approach may be of use in automatic traffic control systems that are fully decentralized and distributed (e.g. traffic-light free roads). As the need for unmanned vehicles grows and their speeds increase, fully decentralized traffic control systems may be the only way to ensure a speedy response to possible collisions without the need to communicate with a centralized controller and the delays involved in such communication. Even if there is an efficient centralized controller, autonomous vehicles will still need some independent

emergency systems to overcome central control failure.

8.2 Previous Work

There has been more interest recently in decentralized approaches to collision avoidance that involve no priority. Distributed reactive collision avoidance [63] is a global deconfliction maneuver which mimics a 'rules of the road' approach such as roundabout passing behaviour. In this approach all vehicles turn the same way until a conflict-free state is reached for the whole system. The approach is 'global' because the positions of all vehicles are required by a deconfliction maintenance controller even if they are not involved in collision, and 'distributed' because collision avoidance computation is distributed among a group of agents. The approach is therefore not autonomous. A collision avoidance approach based on Bernstein-Bézier curves [123] is a cooperative method, where reference-tracking control is used to direct the agents by minimizing the difference in a future trajectory and using the deviation of an agent from its reference path. In this case all agents have a global view and change their paths cooperatively to achieve their individual goals. The limitation of these two approaches is that all-to-all information is needed, where one agent has to consider all other agents' information. Although decentralized in terms of control, these two approaches require a constantly updated global data structure or map for each agent, containing the positions of all other agents. Reciprocal velocity obstacle [135] is a velocity-based collision avoidance approach that lets an agent take just half the responsibility for avoiding collision, while assuming the other agent reciprocates by taking care of the other half. However, the reciprocal velocity obstacle approach

can cause agents to select oscillating velocities as a result of not reaching agreement on which sides to pass each other. The latest work (hybrid reciprocal velocity obstacle [124]) reduces the possibility of oscillations but introduces priority. Knepper and Rus [61] proposed a sampling-based approach with inspiration from human pedestrians, but the approach allows communication between agents and agents also broadcast their own latest planned trajectory to the coordinator. In addition, Toll et al. [138] proposed a multi-layered path planning algorithm for collision avoidance at an airport or a multi-storey building, but again the algorithm is based on global information being available to each agent. Kato et al. [57] outline a traffic rule system for collision avoidance between multiple mobile agents. However, their approach is restricted to route networks, where traffic follows predictable connections between nodes. It is not certain how their approach can be generalized to a complex and large configuration space, where the number of connections can grow exponentially. The approach closest to the one adopted here is [107], where a fully flyable tangential roundabout (FFTR) maneuver is used to prevent collision between two aircraft. However, FFTR requires communication between the two aircraft to agree on the roundabout center and assumes fixed entry and exit points.

Previous work in collision avoidance also does not always take obstacles into account. Obstacles can be fixed (blockages on a straight-line path to a goal) or dynamic (a moving agent getting in the way of another moving agent is an obstacle, and vice versa). The proposed approach takes both sets of potential obstacle into account. Nature-inspired computing can be a major source of inspiration for improving the designs of autonomous agents and robotics. The approach described below is nature-inspired as it applies metrics and processes taken from the area of human kinematics

science (for a review, see [92, 93]) to address collisions between autonomous agents. This research aims to develop and implement a human-like autonomous collision avoidance approach for multi-agent systems using non-priority, local views and a configuration space where an agent can make any move it likes as long as there is no obstacle (fixed or dynamic) in the way. In such a configuration space there are no predetermined routes. Every space in the configuration space is reachable from every neighbouring space provided there is no obstacle.

8.3 Preliminaries

8.3.1 Collision Avoidance Through The Minimal Predicted Distance

Olivier et al. [92, 93] proposed a new minimal predicted distance metric to investigate collision avoidance between two pedestrians. Given two persons with positions p_i and p_j, for $i, j = 1, 2, i \neq j$, each person is considered as a moving obstacle for the other. At each instant t, $MPD(t)$ represents the distance at which a person would meet the other if they did not perform motion adaptation after instant t. According to the model of MPD [92], the future trajectory for each person is modeled as follows:

$$p'(t, u) = p(t) + (u - t)v(\vec{t}), \tag{8.3.1}$$

where u is a future time instant with $u, t > 0$ and $u > t$, $p(t)$ and $v(\vec{t})$ are the position and velocity at time instant t, respectively. Their experimental studies showed that MPD is constant and that walkers adapt their motion only when MPD is small. Therefore, we can predict potential collisions by computing the absolute distance

between p_i and p_j at each time instant t:

$$MPD(t) = \min_{u} \left\| p_i'(t,u) - p_j'(t,u) \right\|. \tag{8.3.2}$$

MPD is a strategy adopted by each robot for predicting potential collision risk. It is also a strategy that attempts to explain how individual humans implicitly adapt their motion and proposes implicit rules that humans naturally and intuitively follow for this adaptation. We further develop this strategy to devise a computational, geometric approach to compute a conflict-free solution for each agent separately and autonomously.

Physical agents will typically calculate paths that suit their own needs. The moves of two or more agents will need to be separated by a minimal safety distance Ω, to ensure no collisions. If two moves along planned paths never take agents within Ω of one another, we say they are conflict-free. That is, paths can intersect but moves along these paths cannot. Put differently, paths can be time-independent but moves along these paths cannot. Formally, moves along paths are conflict-free if and only if

$$\forall t, \forall p_i, p_j, i \neq j, MPD_{ij}(t) > \Omega, \tag{8.3.3}$$

where $MPD(p_i(\vec{v_i}, t), p_j(\vec{v_j}, t))$ is the Euclidean distance between two positions at each time step, and Ω is the grid size dynamically adapted to the configuration space.

8.3.2 Collision and Conflict Definition

The agents considered here are modeled as point masses. However, physical agents have actual size constraints and we need to take physical size into account in the theoretical model. In [76], we investigated all possible collision types between two

moving agents in a configuration grid space, where the collision avoidance condition is to not occupy the same position during the same time-step when following paths, but rather to keep moving within a minimal safety distance at all times. This minimal safety distance has been studied in [92, 93] and is considered a useful metric for minimal predicted distance. Collision can be defined as follows:

Definition 1 (Collision State)**.** A collision occurs between agents A_i and A_j when

$$\mathcal{C}_{ij} \Leftrightarrow \|p_i - p_j\| < \Omega(A_i, A_j), \tag{8.3.4}$$

where $\mathcal{C}_{ij}$ represents the collision between two agents A_i and A_j, Ω is a distance threshold for the minimal safety distance, which in turn is the absolute distance between the agents' geometric centers. Thus, we have the non-collision state description as follows:

Definition 2 (Non-Collision State)**.**

$$\mathcal{S}_{ij} \Leftrightarrow \|p_i - p_j\| \geq \Omega(A_i, A_j), \tag{8.3.5}$$

where $\mathcal{S}_{ij}$ represents the non-collision state of the two agents corresponding to $\mathcal{C}_{ij}$ condition.

Definition 3 (Conflict State)**.** Another situation that must be accounted for is when collision would occur if two agents do not perform motion adaptation at a future time instant t. According to Equation 8.3.2, a conflict occurs between agents A_i and A_j if the agents are not currently in a collision situation but will enter a collision situation at time u if they do not perform motion adaptation. Equation 8.3.6 gives the definition of this conflict:

$$\mathcal{H}_{ij}(t) \Leftrightarrow \mathcal{S}_{ij}(t) \wedge MPD(u) < \Omega(A_i, A_j), \tag{8.3.6}$$

where $\mathcal{H}_{ij}(t)$ represents conflict between two agents A_i and A_j at time instant t taking into account the future time u (Equation 8.3.2). $\wedge$ is the conjunction operator.

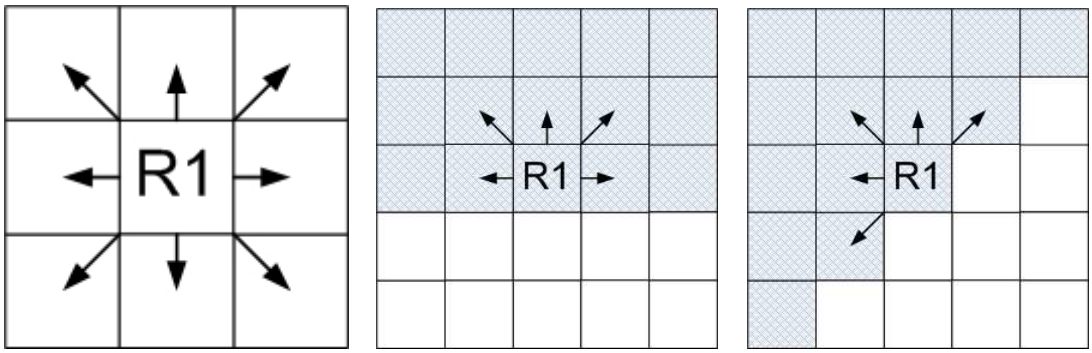
8.4 Deconflict Through MER Roundabout Method

The proposed MER roundabout method is a pairwise collision avoidance maneuver and includes two phases – a conflict detection phase and a deconflict phase.

8.4.1 Local View Definition

We define a local view LV in front of the current position of an agent and only take into account the agents and any other obstacles inside this local view. The local view has to be of a minimum size to ensure satisfactory conflict detection. If the configuration space is considered as consisting of a grid of squares or rectangles, the size of which is equal to the size of the agent, each agent has 8 moving directions at each time step, as shown in Figure 8.1(a) and a wait action, plus front local view, as shown in Figure 8.1(b) and (c). All agents have a constant speed for simplicity. My approach requires each agent to consider its moves within its front local view at each time step, so each agent potentially has 9 legal actions. Each of these legal actions is a solution to the constraint satisfaction problem in which each agent must determine a move from $\{E,S,W,N,NE,SE,SW,NW,wait\}$, as shown in Figure 8.1(a), provided that the chosen move does not lead to collision with another agent.

The front local view will be restricted to the region that the agent can actually see, given the direction of motion of the agent, its view angle, and the position of any static obstacles (and perhaps other agents). The LV needs to be updated once the new velocity $\vec{v}$ is computed. Fixed and dynamic obstacles will be presented in the LV of each agent, not in a global data structure to be shared by all agents.



(a) (b) (c)

Figure 8.1: (a) 8 possible moving directions. (b) and (c) The front local view (LV) of the agent.

8.4.2 Deconflict Maneuver

Given an agent A_i and n number of neighbour agents $A = \{A_1, A_2, ..., A_n\}$ with $1 \leq j \leq n$ in LV, if two agents' moves conflict $\mathcal{H}_{ij}$, a virtual rectangular roundabout can be computed by calculating a minimum enclosing rectangle,

$$R^{ij} = MER(p_i, p_j), \tag{8.4.1}$$

where $p_i, p_j \in R^{ij}, \eta \equiv 2$. That is, the boundary of the rectangle is also included in the rectangle. Then, a new velocity is calculated over R^{ij}.

To calculate the new velocity $\vec{v}$ over R^{ij} for deconfliction between A_i and A_j, we calculate the other two diagonal opposite corner points p_i' and p_j', and we have

$$\mathrm{p}' = \begin{cases} q_1 & \min\{x_i, x_j\} \; and \; \min\{y_i, y_j\}, \\ q_2 & \min\{x_i, x_j\} \; and \; \max\{y_i, y_j\}, \\ q_3 & \max\{x_i, x_j\} \; and \; \min\{y_i, y_j\}, \\ q_4 & \max\{x_i, x_j\} \; and \; \max\{y_i, y_j\}. \end{cases} \tag{8.4.2}$$

Here, p_i, p_j correspond to two distinct elements of p′. Then we have another two points p_i' and p_j'

$$\{p_i', p_j'\} = \mathrm{p}' \cap \neg\{p_i, p_j\}. \tag{8.4.3}$$

Some elementary properties of rectabout maneuver that we will use in this research are introduced as follows:

Lemma 1 (Symmetry)**.**
$\vec{v_i} \in MER(p_i, p_j) \Leftrightarrow \vec{v_j} \in MER(p_j, p_i)$,

Symmetry property follows from Figure 4.2. The velocities $\vec{v_i}$ and $\vec{v_j}$ belong to the boundary of the rectabout (see Figure 8.2).

Lemma 2 (Keep Left Traffic Rule)**.**
$p_i' \in MER(p_i, p_j) \wedge p_j' \in MER(p_j, p_i) \Leftrightarrow \Delta p_i p_j p_i' > 0 \wedge \Delta p_i p_j p_j' < 0.$

Proof. Construct three vertices p_i, p_j, p_i' and p_i, p_j, p_j' to form two triangles. According to the vector cross product theory [81], the area of a triangle Δ can be calculated using the cross product to keep track of the signs of each angle, i.e., by the sign of the expression

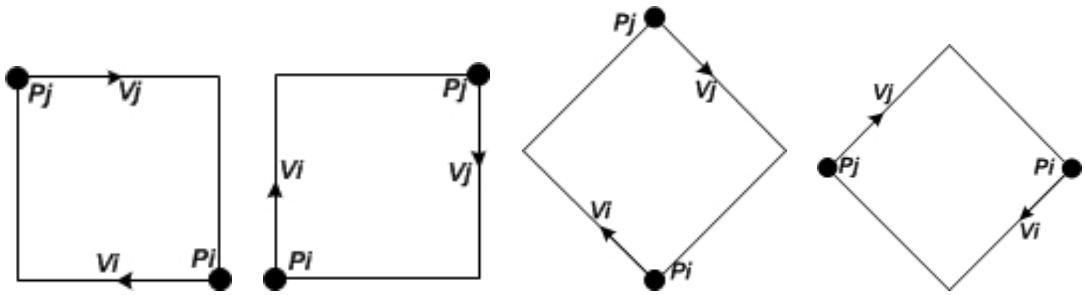


Figure 8.2: The symmetry property of the rectabout maneuver for collision avoidance. The velocities $\vec{v_i}$ and $\vec{v_j}$ are the new velocities after deconfliction by MER rectabout.

$$\Delta = \begin{cases} \frac{1}{2}((x_i - x_i')(y_j - y_i') - (y_i - y_i')(x_j - x_i')), \\ \\ \frac{1}{2}((x_i - x_j')(y_j - y_j') - (y_i - y_j')(x_j - x_j')). \end{cases} \tag{8.4.4}$$

Since the 'keep left' traffic rule is used in this work, we always select an anti-clockwise point as a solution from the two diagonally opposite corner points p_i' and p_j'. Clockwise motion is represented by a negative value and anticlockwise by a positive value. Therefore:

$$\mathrm{p}' = \begin{cases} p_i' & \Delta > 0 \quad \text{anticlockwise - keep left traffic rule,} \\ p_j' & \Delta < 0 \quad \text{clockwise - keep right traffic rule.} \end{cases} \tag{8.4.5}$$

□

The new velocity $\vec{v_i'}$ can be calculated as

$$p_i' - p_i = (x_i' - x_i) \wedge (y_i' - y_i). \tag{8.4.6}$$

For $x < 0$,

$$\vec{v_i'} = \begin{cases} (-1,-1) & \text{if } y < 0, \\ (-1,0) & \text{if } y = 0, \\ (-1,1) & \text{if } y > 0. \end{cases}$$

For $x = 0$,

$$\vec{v_i'} = \begin{cases} (0,-1) & \text{if } y < 0, \\ (0,1) & \text{if } y > 0. \end{cases}$$

For $x > 0$,

$$\vec{v_i'} = \begin{cases} (1,-1) & \text{if } y < 0, \\ (1,0) & \text{if } y = 0, \\ (1,1) & \text{if } y > 0. \end{cases}$$

Assuming the system consists of n neighbour agents, this algorithm's computation time on each agent scales as $O(n)$, since it only requires the computation of each agent's move to detect conflict and then compute the MER for deconfliction. Formally, $O(n^2)$ MER computations occur in the worst case, but since these computations are independent of each other, they are calculated in parallel in a distributed fashion, so only the per-agent scaling matters. However, to calculate the performance of a system of multiple agents for comparison with other approaches we need to assume a centralized control strategy version of this distributed conflict detection and deconfliction strategy (Results, below).

8.4.3 Guarantees

We now prove that the MER Rectabout can be used to generate conflict-free, deadloop-free [1] and deadlock-free motions for each agent.

[1]Deadloop problem is defined in [76]. Oscillation problem [136] is similar to the deadloop and is considered as deadloop in this work.

Conflict-Free Navigation

Let $\vec{v}_A$ be the current velocity of agent A, and let $\vec{v}_B$ be the current velocity of agent B, and let both A and B choose new velocities ($\vec{v}'_A$ and $\vec{v}'_B$) based on the other two diagonal opposite corner points p'_A and p'_B. The following theorem proves that this is safe, provided that both agents choose the same side (keep left) to pass each other:

Theorem 1 (Conflict-Free)**.**

$\vec{v}'_A \in MER(p_A, p_B) \wedge \vec{v}'_B \in MER(p_B, p_A) \Rightarrow \vec{v}'_A \times \vec{v}'_B = 0$

Proof.

$MER(p_A, p_B) \wedge MER(p_B, p_A)$

$\Rightarrow$ {Equation 8.4.1 and Equation 8.4.3}

$p'_A \in MER(p_A, p_B) \wedge p'_B \in MER(p_B, p_A)$

$\Leftrightarrow$ {Equation 8.4.6 and Lemma 2}

$\vec{v}'_A \in MER(p_A, p_B) \wedge \vec{v}'_B \in MER(p_B, p_A)$

$\Rightarrow$ {Lemma 2}

$\vec{v}'_A \times \vec{v}'_B = 0$

$\Leftrightarrow$ {Equation 8.3.3 and Definition 3}

$MPD(p_A(\vec{v}'_A, t), p_B(\vec{v}'_B, t)) > \Omega(A, B)$ □

Deadloop-Free Navigation

If all agents follow the same 'keep left' rule, this guarantees deadloop-free navigation. This is proven by the following theorem:

Theorem 2 (Deadloop-Free)**.**

$\vec{v}'_A \in MER(p_A, p_B) \Leftrightarrow \Delta p_A p_B p'_A > 0 \wedge \vec{v}'_B \in MER(p_B, p_A) \Leftrightarrow \Delta p_B p_A p'_B > 0$

Proof.

$\vec{v'}_A \in MER(p_A, p_B) \wedge \vec{v'}_B \in MER(p_B, p_A)$

$\Leftrightarrow$ {Equation 8.4.6}

$p'_A \in MER(p_A, p_B) \wedge p'_B \in MER(p_B, p_A)$

$\Leftrightarrow$ {Theorem 1}

$\vec{v'}_A \times \vec{v'}_B = 0$

$\Leftrightarrow$ {Lemma 2}

$\Delta p_A p_B p'_A > 0 \wedge \Delta p_B p_A p'_B > 0$ □

Deadlock-Free Navigation

Let the conflict group, $\mathcal{S}_\mathcal{H}$, contains n constant-speed agents that are in conflict and deadlock. Let m agents be in deadlock in $\mathcal{S}_\mathcal{H}$ where no solution is found and choose the 'wait' action, $m, n \in \Re, n > m$. We guarantee deadlock-free navigation if and only if $n > m$. This is the case even though these conflicts can be daisy-chained such that, if each agent is in conflict with another, this analysis still guarantees that at least one agent in the group can find a move solution. Hence, some agents must eventually attain their goal and allow another to progress, thereby breaking the deadlock. This is proven by the following theorem:

Definition 4 (Deadlock State)**.** We consider the deadlock state if the new velocity belongs to a set of conflict velocities for the current agent A_i:

$\vec{v'_i} \in \mathcal{S}_\mathcal{H}(A_i, A_j)$,

where $\mathcal{S}_\mathcal{H}(A_i, A_j)$ is a set of conflict velocities for the current agent A_i with neighbour agents A_j, denoted as $\mathcal{S}_\mathcal{H}(A_i, A_j) = \{\vec{v_i} | \vec{v_i} \in MER(p_i, p_j)\}, i \neq j, j \in n$. The $\mathcal{S}_\mathcal{H}$ is always updated once the new velocity $\vec{v'}$ is computed for each agent.

Theorem 3 (Deadlock-Free)**.**

A_i, A_j are in conflict, $\vec{v_i}, \vec{v_j} \in \mathcal{S}_{\mathcal{H}}, i \in m, j \in n, n > m \Rightarrow \left\|\vec{v_i'}\right\| \neq 0$.

Proof.

A_i, A_j are in conflict

$\Rightarrow$ {Definition 8.3.6}

$\mathcal{H}_{ij}(t)$

$\Rightarrow$ {Equation 8.4.1}

$p_i' \in MER(p_i, p_j) \wedge p_j' \in MER(p_j, p_i)$

$\Leftrightarrow$ {Lemma 1}

$\vec{v_i'} \in MER(p_i, p_j) \wedge \vec{v_j'} \in MER(p_j, p_i)$

$\Rightarrow$ {Theorem 1 and Definition 4}

$\left\|\vec{v_i'}\right\| = 0 \wedge \left\|\vec{v_j'}\right\| \neq 0$ at time instant t

$\Rightarrow$ {Theorem 1 and Definition 4}

$\left\|\vec{v_i'}\right\| \neq 0$ at time instant $t+1$ □

8.4.4 Rectabout Algorithm

Given $n > 0$, the rectabout is calculated as the following at each time step:

1. Estimate the motion state pairwise for agent A_i with neighbour agents A_j in LV of velocity $\vec{v}$ by $MPD(A_i, A_j)$ where $i, j \in n$, $i \neq j$,

2. Repeat until there is no conflict: If there is conflict, then compute R^{ij} by $MER(p_i, p_j)$. If no solution is found, then execute the wait action and terminate; otherwise

3. Find the other two diagonally opposite corner positions p_i' and p_j',

4. Calculate the new velocity $\vec{v_i'}$ using the two points p_i and p_i'.

5. Update the information of neighbour agents and static obstacles in LV of new velocity $\vec{v'}$.

This procedure is repeated until the deconfliction motion is found (including a wait action) for all the neighbour agents. Figure 8.3 illustrates agent 1 (R1) computing pairwise virtual rectabouts to avoid collisions with two other agents R2 and R3 (a). First, R1 calculates a rectabout to avoid R2 and plans a move NW (b). However, R3 is also in conflict (c), so R1 calculates another rectabout and planned move W to avoid R3 (d). Similarly, the other two agents use the same procedure to deconflict.

8.5 Experimental Results

We performed both small-scale simulations to test local behaviour and large-scale simulations to analyze performance scaling. We also compared the proposed approach to a centralized approach. The simulations model used mobile agents of constant size and speed, where the task is to move from their current position towards a goal position. Each agent has its own random current and goal positions. Paths calculated by agents can intersect and in certain move situations lead to conflict. The aim is to minimize stop and wait states so that agents can reach their goal in the minimum time while avoiding collision.

8.5.1 Local Behavioural Results

We show two scenarios which demonstrate how agents smoothly avoid collisions with each other at the local level. In the first scenario shown in Figure 8.4(a), two agents

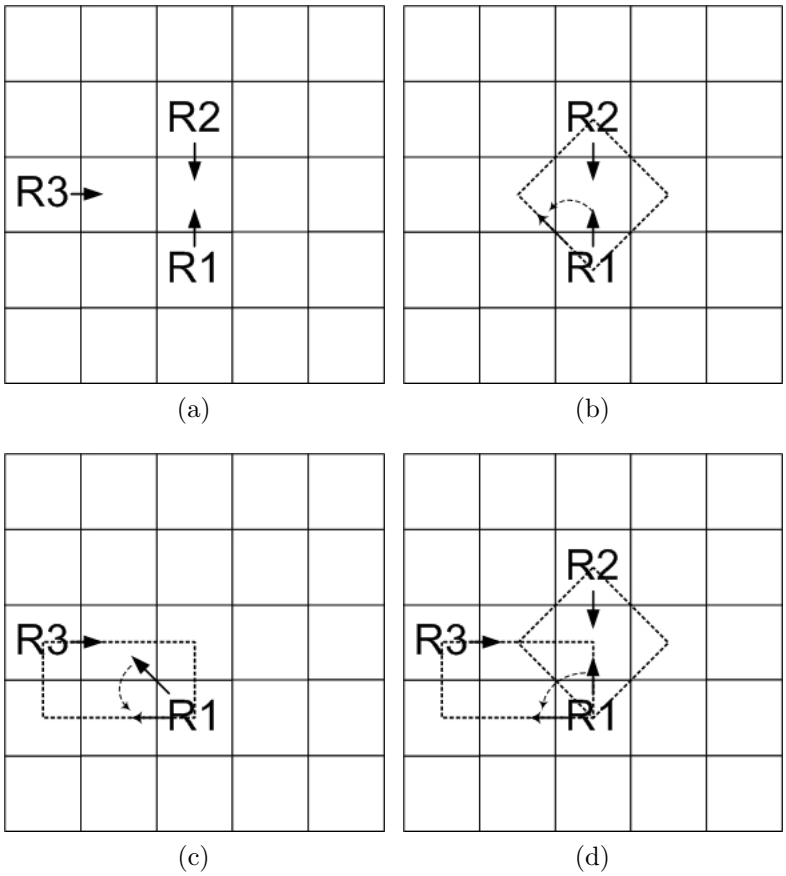
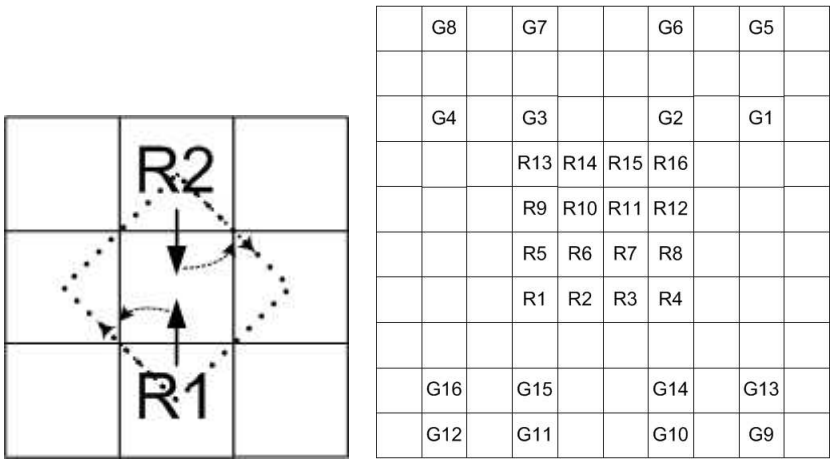


Figure 8.3: Illustration of how rectabouts resolve conflicts between three agents. Agent 1 computes virtual rectabouts by pairwise approach based on MER for deconfliction.

are moving towards each other. Each agent is able to detect conflict with the other and computes a virtual rectabout based on MER ($MER(p_1, p_2)$). A new velocity is planned along the path of MER and the agents follow a shared 'keep left' traffic rule to resolve the conflict independently. The virtual rectabout is removed after one time-step, after which each agent needs to independently operate the process again, since the information around the agent always changes with every time step. However, agents always attempt to move towards their own goal position at every time-step.

The second scenario involves 16 agents where all agents are densely located in the center of the environment, shown in Figure 8.4(b). In this worst-case scenario, using rectabouts may cause deadlock and the wait action (one timestep) is used for the agents involved in deadlock. Agents will follow their goal-direction again after one time step. An agent takes avoiding action irrespective of whether an obstacle in its LV is fixed (non-moving agent) or dynamic (moving robot). Videos of these scenarios can be found at `http://youtu.be/GSH_i98Ju6w`.



(a) (b)

Figure 8.4: Two small behavioural simulations. R and G represent Agent and Goal for each agent, respectively. (a) The solid arrow line is the intended trajectory. The dotted arrow line is the deconfliction trajectory. The central dotted rectangle is a virtual rectabout enclosing two agents R1 and R2. (b) The 16 agents are densely located in a 10x10 grid environment. Each agent moves to its antipodal position in the environment, leading to maximum possible conflict and possible deadlock.

8.5.2 Large Scale Simulation Results

In order to test the performance of the proposed method we varied the number of agents in different configuration spaces (200x200 grid, 300x300 grid and 400x400 grid)

to see how our approach scales when the number of agents increases. We performed my experiments on an Intel Core(TM) i5 processor 3.20 GHz with 4GByte of memory. Each scenario was repeated 10 times and results were averaged. All start and goal positions were generated randomly. Figure 8.5 shows the total running time for various numbers of agents. We note that the total running time of the proposed method scales nearly linearly with the number of agents. Furthermore, the computation time increases as the density of agents increases, as would be expected given that deadlock is more frequent with high density.

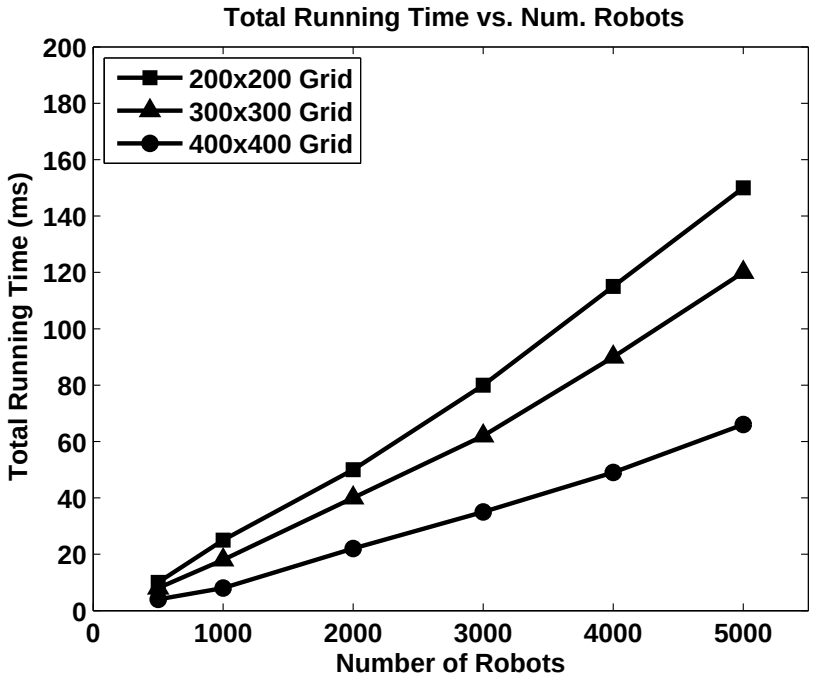


Figure 8.5: The total running can be seen to scale almost linearly with the number of agents.

8.5.3 Additional Case Studies

The proposed approach has been tested in extensive simulations to evaluate the robustness of the collision avoidance system, using multiple autonomous agents with different goals that involve crossing trajectories. Three case studies are presented below. The robustness of collision avoidance is evaluated on: (A) the capability of the agent's position, i.e. the system is not restricted to any synchronous, symmetric and communication-based maneuvers; (B) the adaptability of collision avoidance in case of deadlock; and (C) the scalability of collision avoidance to increasingly larger numbers of agents / robots.

Case Study 1: Collision Avoidance Capability

Figure 8.6 (a) depicts three agents R1, R2 and R3 moving to their goal position G1, G2 and G3, respectively. Figure 8.6 (b) presents R1's trajectory being in conflict with R2 and R3's trajectories because their paths cross each other (i.e. $\mathcal{H}_{12}$ and $\mathcal{H}_{13}$). R1 computes rectabouts corresponding to R2 and R3 conflicts. R1 re-plans a moving direction for only one move. After one move, R1 resumes its goal-directed path and repeats the process. Similarly, R2 and R3 follow the same procedure. Figure 8.6 (c) shows R2 and R3 employing a similar approach to change their original moving directions to avoid each other.

Case Study 2: Collision Avoidance Adaptability

We conduct an experiment with five mobile agents, where one agent R1 cannot find a solution. Figure 8.7 (a) shows R1 is surrounded by its four neighbour agents. In this case R1 using MER roundabout may cause deadlock, as shown in Figure 8.7

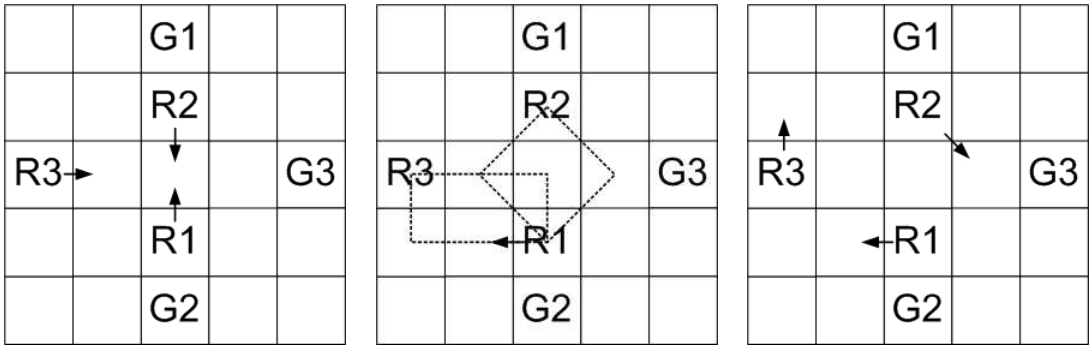


Figure 8.6: Collision avoidance for 3 agents. (R1, R2, R3) and (G1, G2, G3) are the agent positions and the goal positions for three agents, respectively. (a) The initial position for three agents. (b) R1 computes MER rectabout and re-plans its moving direction in order to avoid collisions with the other two neighbour agents. (c) The other two agents employ a similar approach to obtain a new moving direction.

(b). The wait action is used while the agent is involved in deadlock in Figure 8.7 (c). Wait only takes one time step. Agents will follow their goal-direction again after one time step. The iterative plan-evaluate-move process runs until all agents reach their goal. As seen in this case study, MER-based rectabout collision avoidance maneuver is adaptable to all possible collisions in [76] and to deadlock.

Case Study 3: Collision Avoidance Scalability

To evaluate the robustness of large scale agent systems, we expand the configuration space to a 50x50 grid involving 10, 20, 50 and 100 agents. All start and goal positions are generated randomly. The proposed algorithms are able to move all agents to their desired destination nodes avoiding collision. Figure 8.8 shows the scalability performance evaluation on 10, 20, 50 and 100 agents by 50x50 grid configuration space. The moves increase linearly as the number of agents increases.

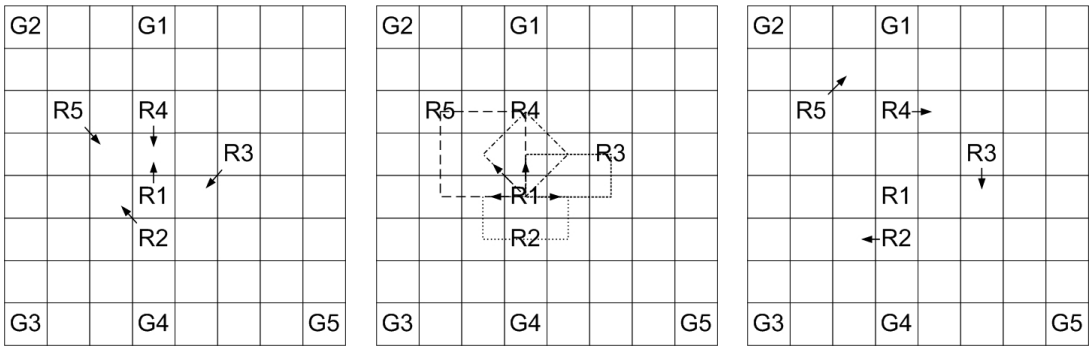


Figure 8.7: Collision avoidance for 5 agents. (a) The initial position for five agents. (b) R1 computes MER rectabout in relation to the other neighbour agents, but cannot find a solution and causes deadlock. (c) R1 takes wait action while in deadlock in such a case. The other agents use a similar approach to compute a new deconfliction moving direction.

8.5.4 Comparison with Centralized Approach

The performance of the MER rectabout maneuver was compared against a centralized priority-based algorithm described in Chapter 5, called Super A*. The test configuration space was based on a 50x50 grid involving 10, 20, 50 and 100 agents. All start and goal positions were generated randomly. The result was evaluated by the total number of moves for all agents moving from their start positions to their goal positions. To undertake this comparison, the proposed decentralized approach was represented as a centralized approach (a central coordinator needing to calculate all LVs for every agent and all collision and collision avoidance strategies). Table 8.1 presents the statistical results for the MER rectabout approach and shows that it is comparable to a centralized approach. The big difference, however, is that Super A* (and other centralized approaches) has costly overheads in terms of communication and global map updates.

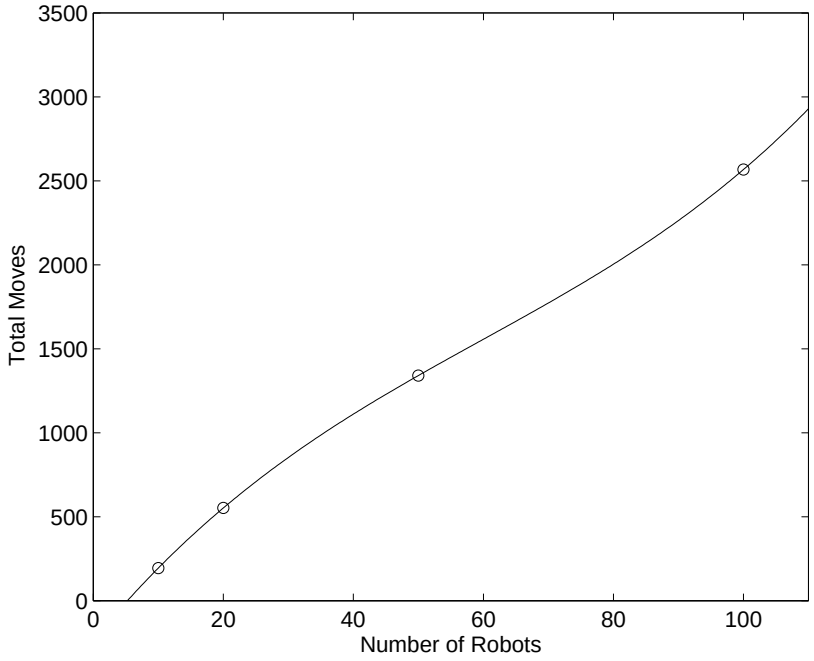


Figure 8.8: Scalability evaluation on 10, 20, 50 and 100 agents by 50x50 grid configuration space. The total moves show a linear increase as the number of agents increases.

8.6 Summary

A novel non-priority and non-global view rectabout approach based on MER for collision avoidance is proposed in this chapter. The kinematic research on the minimal predicted distance between two human walkers is applied for the first time to deal with agent collision problems in conjunction with a rectabout maneuver. We use MPD to detect the possible collisions along agent paths and trajectories. The agents involved in conflict will compute a rectabout and re-plan a new velocity when MPD is below the threshold. This process is repeated until all agents reach their goals. The proposed approach also takes into account obstacles and sudden changes of direction

Number of Agents	10	20	50	100
Super A*	194	543	1318	2554
MER rectabout	194	552	1341	2568

Table 8.1: MER rectabout compared with a centralized priority-based approach. Performance evaluation on the total number of moves for 10, 20, 50 and 100 agents by 50x50 grid configuration space.

by agents.

While this work presents a significant advancement, there are several issues which require further work. One key area for future work is the effect on MPD and MER of variable sized local views. The second is the need for dealing with variable speed agents. Also, while the aim of this work is to remove the need for any communication between agents, there may be occasions (e.g. deadlocks, cooperative behaviour) where some form of communication is required or desired.

Finally, while the work has focused on collision detection and avoidance among agents and has presented simulations involving agents of fixed size and speed, the research needs extending to problem solving by a number of agents needing to search a complex space cooperatively as well as autonomously. In particular, it is not clear how generalizable the use of rectangles will be in more complex spaces. Nevertheless, it could be argued that the most important aspect of any mobile multi-agent system is how to prevent agents from getting in each other's way. Such collision avoidance is often overlooked or ignored in mobile multi-agent and swarm-based approaches and simulations. Swarm simulations frequently assume that swarm members can fly through each other. The lack of collision detection and avoidance severely limits the application of swarm technology to real-life problems (e.g. nanobots taking antiviral molecules to specific cells in the body).

To conclude this part dealing with fully autonomous multi-agent system, we have shown that, how nature-inspired observations help to resolve the research question / issue for collision avoidance problem. In the previous chapters, we eliminated coupling priority, decoupling priority, communication, and global views which are essential aspects in multi-agent systems step by step, in order to explore a research procedure for nature-inspired autonomous, non-centralized, non-global, non-communicative collision avoidance. We have proved that our proposed approach is guaranteed and reliable for safety. In the next chapter, we will expand the idea of rectabout to a heterogeneous multi-agent system for collision avoidance.

Chapter 9

Intelligent Collision Avoidance between Multiple Autonomous Hybrid Agents using Adaptive Local Views

The aim of this chapter is to address research question 5 (Research Methodology chapter): Can a large number of homogeneous autonomous agents extend to heterogeneous agents with variable size and speed? This research question follows naturally from the fourth research question identified in the Research Methodology chapter. The fourth research question led to the novel solution of a nature-inspired autonomous collision avoidance – MPD for detection of collisions and MER-based rectabout for deconfliction. But the question was left open in the previous chapter as to how to extend the homogeneity modelling to heterogeneity modelling, which allows agents to have variable size and speed.

The previous chapter presents an autonomous multi-agent system; the idea from nature – the use of roundabouts – describes deconfliction between two vehicles with no priority, no communication and local view. The purpose of this chapter is to expand a homogeneous to a heterogeneous system, which introduces a hybrid rectabout modeling

for a group of heterogeneous agents in collision avoidance problems. Section 1 briefs the idea of the hybrid rectabout. In Section 2, we formally define the problem we address in this chapter; we introduce minimal predicted distance for conflict detection used in our approach, and also provide definitions of collision avoidance. Section 3 introduces our formulation of hybrid rectabout collision avoidance maneuver for various speed multiple autonomous agents; it also takes into account obstacles in the environment as well as uncertainty in position and velocity, and the dynamics and kinematics of the agents. We discuss implementation and present experimental results in Section 4. Finally, section 5 gives the conclusion of this work and some further research directions.

9.1 Introduction

In this chapter, the main contribution focuses on two aspects. First, we present a hybrid rectabout procedure for navigation of multiple mobile robots or virtual agents that explicitly considers heterogeneity (i.e. variable speed and variable size of agents). Second, we present both practical demonstrations and experimental simulations to demonstrate scalability and efficiency, taking into account Minimal Predicted Distance (MPD, more details below) between all the autonomous agents and the velocity of each agent. The approach is 'nature inspired' because of its use of theories taken from human pedestrian collision avoidance. The approach is 'intelligent' because of its use of traffic regulations and conventions. It is also fully decentralized, with each agent taking responsibility independently for detecting and avoiding collisions through local views / maps and local decision-making.

The hybrid rectabout is an extension of the Minimum Enclosing Rectangle-based

(MER-based) rectabout described in Chapter 8 that was introduced to address similar issues in multiagent simulation. However, the rectabout formulation had some limitations. All agents were required to be homogeneous (same size and speed). This meant that all agents had the same local view. To extend the applicability of the MER-based rectabout to heterogeneous agents, the procedure needs to take into account variable speeds of agents as well as variable size of agents. Both require adaptive local views since variable speeds of agents means not just that agents are moving at different speeds from each other but also that an agent can vary its own speed while moving. The size of an agent is, however, constant. The aim of this chapter is to extend the MER-based rectabout procedure to deal with heterogeneous agents (variable speeds of agents and various sizes of agents), hence the term 'hybrid'. To deal with this hybrid nature of agents the adaptive MER-based rectabout procedure to be described below is also hybrid in that the size and location of the rectabout will vary to reflect the properties of the agent. Consequently, MPD is adaptive to change while the agent speed is variable. In addition, our approach takes into account both the kinematic constraints of an agent and sensor uncertainty, which makes it specifically suitable for navigation of autonomous agents.

Informally, the hybrid rectabout procedure to be described below builds on the implicit assumption that other agents make similar collision avoidance reasoning via MER. That is, knowledge of MER is shared by all agents. It consists of two components: Minimal Predicted Distance (MPD) detection and hybrid rectabout collision avoidance algorithm. The MPD is a metric inspired by real human pedestrian collision avoidance behaviour (for a review, see [91, 92, 93] and more details below). As

far as we are aware, this was the first time that MPD has been used for addressing collision problems in multi-agent systems. The hybrid MER-based rectabout collision avoidance algorithm is a pairwise approach which computes and navigates agents' moving direction by following a 'keep right' rule at the rectabout. This rule can be changed if necessary to 'keep left'.

We have implemented and applied our new approach for heterogeneous agents to a set of WowWee Rovio mobile robots moving in an indoor environment using independent sensing and WiFi-based wireless remote control. Our experiments show that our approach achieves direct, collision-free and oscillation-free navigation in an environment containing multiple mobile robots and dynamic obstacles, even with some uncertainty in position and velocity. We also demonstrate the ability to handle static obstacles and the low computational requirements, scalability and efficiency of the hybrid rectabout in simulations of multiple virtual agents.

9.2 Problem Definition

In this section, we introduce the collision issues in multi-agent systems and collision avoidance through a minimal predicted distance concept in our approach.

9.2.1 Collision Issues

The problem we discuss in this chapter is formally defined as follows. Let there be a set of n agents sharing an environment. For simplicity we assume the agents move in the plane $\mathbb{R}^2$. Each agent A has a current position p_A, a current velocity $\vec{v_A}$. An agent's position can be obtained through sensors on the agent and the information

can be broadcast through a WiFi-based remote control if necessary. In other words, all we need to demonstrate our new approach is that an agent can observe another agent when it 'comes into view' and that every agent knows what its position is in relation to the configuration space. These parameters are part of the agent's external state. Furthermore, each agent can have a different speed while moving from start location to goal location, for instance, starting slow and speeding up.

The task is for each agent A to independently and simultaneously calculate a new velocity $v_A^{\vec{new}}$ for itself such that, at an emergent level, all agents are guaranteed to be conflict-free for at least a certain amount of time (one time step in our experiments) when they would continue to move at their new velocity. As a secondary objective, each agent should calculate its new velocity as close as possible to its goal orientation so that, at an emergent level, all agents reach their goal. The agents are not allowed to negotiate with each other, and can only use observations of the other agent's current position and velocity. However, each of the agents may assume that the other agents use the same strategy as itself to select a new velocity.

9.2.2 Minimal Predicted Distance

As seen in the last chapter, inspiration from nature comes from Olivier et al. [92, 93], who proposed a new minimal predicted distance metric to investigate how pedestrians effortlessly and without communication avoid each other repeatedly and in a variety of different circumstances while still reaching their goals with minimum disruption to their paths. To recap, given two persons with positions p_i and p_j, for $i, j = 1, 2, i \neq j$, each person is considered as a moving obstacle for the other. At each instant t, $MPD(t)$ represents the distance at which a person would meet the other if they did

not perform motion adaptation after instant t. According to the model of MPD [92], the future trajectory for each person is modeled as follows:

$$p'(t,u) = p(t) + (u-t)v(\vec{t}), \tag{9.2.1}$$

where u is a future time instant with $u, t > 0$ and $u > t$, $p(t)$ and $v(\vec{t})$ are the position and velocity at time instant t, respectively. Their experimental studies showed that MPD is constant and that walkers adapt their motion only when MPD is small. Therefore, we can predict potential collisions by computing the absolute distance between p_i and p_j at each time instant t:

$$MPD(t) = \min_{u} \left\| p_i'(t,u) - p_j'(t,u) \right\|. \tag{9.2.2}$$

MPD is a strategy adopted by each agent for predicting potential collision risk. It is also a strategy that attempts to explain how individual humans implicitly adapt their motion and proposes implicit rules that humans naturally and intuitively follow for this adaptation. We further develop this strategy to devise a computational, geometric approach to compute a conflict-free solution for each agent separately and autonomously.

The further effects on MPD for two pedestrians walking at different speeds are revealed in [91], where computing the MPD with respect to motion adaptation shows the extent to which MPD is adapted when the speed s or orientation θ of two walkers varies. We formalize this as:

$$MPD_{ij}(t) = f(p_i(t,u), p_j(t,u), \theta_i, s_i, \theta_j, s_j), \tag{9.2.3}$$

Physical agents will typically calculate paths that suit their own needs. The moves

of two or more agents will need to be separated by a minimal safety distance, Ω, to ensure no collisions. If two moves along planned paths never take agents within Ω of one another, we say they are conflict-free. That is, paths can intersect but moves along these paths cannot. Put differently, paths can be time-independent but moves along these paths cannot. Formally, moves along paths are conflict-free if and only if

$$\forall t, \forall p_i, p_j, i \neq j, MPD_{ij}(t) > \Omega, \tag{9.2.4}$$

where $MPD_{ij}(t)$ is the Euclidean distance between two positions at each time step, and Ω is the grid size dynamically adapted to the configuration space to compute the minimal safety distance.

9.2.3 Collision and Conflict Definition

To recap, the agents considered here are modeled as point masses. However, physical agents have actual size constraints and we need to take physical size into account in the theoretical model. Liu et al. [76] investigated all possible collision types between two moving agents in a configuration grid space, where the collision avoidance condition is to not occupy the same position during the same time-step when following paths, but rather to keep moving within a minimal safety distance at all times. This minimal safety distance has been studied in [91, 92, 93] and is considered a useful metric for minimal predicted distance. Collision can be defined as follows:

Definition 5 (Collision State)**.**

A collision occurs between agents A_i and A_j when

$$\mathcal{C}_{ij} \Leftrightarrow \|p_i - p_j\| < \Omega(A_i, A_j), \tag{9.2.5}$$

where $\mathcal{C}_{ij}$ represents the collision between two agents A_i and A_j, Ω is a distance threshold for the minimal safety distance, which in turn is the absolute distance between the agents' geometric centers. Thus, we have the non-collision state description as follows:

Definition 6 (Non-Collision State)**.**

$$\mathcal{S}_{ij} \Leftrightarrow \|p_i - p_j\| \geq \Omega(A_i, A_j), \tag{9.2.6}$$

where $\mathcal{S}_{ij}$ represents the non-collision state of the two agents corresponding to $\mathcal{C}_{ij}$ condition.

Definition 7 (Conflict State)**.**

Another situation that must be accounted for is when collision will occur if two agents do not perform motion adaptation at a future time instant t. According to Equation 9.2.2, a conflict occurs between agents A_i and A_j if the agents are not currently in a collision situation but will enter a collision situation at time u if they do not perform motion adaptation. Equation 9.2.7 gives the definition of this conflict:

$$\mathcal{H}_{ij}(t) \Leftrightarrow \mathcal{S}_{ij}(t) \wedge MPD(u) < \Omega(A_i, A_j), \tag{9.2.7}$$

where $\mathcal{H}_{ij}(t)$ represents conflict between two agents A_i and A_j at time instant t taking into account the future time u (Equation 9.2.2). '$\wedge$' is the conjunction operator.

9.3 Collision Avoidance

In this section, we describe how agents avoid collisions with each other. We briefly review the idea of MER [23], and then introduce our formulation of the hybrid MER-based rectabout that we use for heterogeneous multi-agent navigation.

9.3.1 MER Representation

According to Das et al. [23], given a set of points $P = \{p_1, p_2, ..., p_k\}$ with $p_k \in \Re^2$, the minimum enclosing square (or rectangle) of P is the smallest square (or rectangle) that contains all points of P. For the purposes of this chapter, the smallest square or rectangle is defined to be the smallest rectangle that contains a given number k such that $\frac{n}{2} < k \leq n$ of x-consecutive points in a set of n points in the plane. The problem of computing *k-square* and *k-rectangle* has been investigated since 1991 (for a review, see [1, 23, 31, 79, 115]). MER has been applied in various areas, such as pattern recognition [96], facility location [29], similarity search [24, 88] and collision detection [75]. In order to classify the *k-square* with respect to the number of points η present on its boundary, Das et al. [23] investigated all different possibilities of *k-squares*. As a result, no *k-square* is possible with $\eta = 0$ or 1. The only possibility with $\eta = 2$ is that the two points appear at the two diagonally opposite corners of the corresponding *k-squares*. In this study, $k = \eta = 2$ is the MER or MES that the agents are searching for, as shown in Figure 9.1.

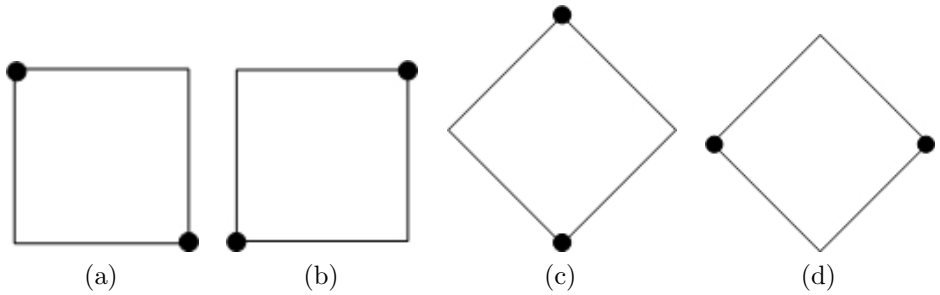


Figure 9.1: MER of $\eta = 2$, with dots representing the position of the two agents. The orientation of the rectabout can differ according to the local view.

9.3.2 Local View Definition

We define a local view (LV) in front of the current position of an agent and only take into account the agents and any other obstacles inside this local view. The local view has to be of a minimum size to ensure satisfactory conflict detection. If the configuration space is considered as consisting of a grid of squares or rectangles, the size of which is equal to the size of the agent, each agent has 8 moving directions at each time step and a wait action, plus front local view. Our approach requires each agent to consider its moves within its front local view at each time step, so each agent potentially has 9 legal actions. Each of these legal actions is a solution to the constraint satisfaction problem in which each agent must determine a move from $\{E,S,W,N,NE,SE,SW,NW,wait\}$, provided that the chosen move does not lead to collision with another agent.

The front local view will be restricted to the region that the agent can actually see, given the direction of motion of the agent, its view angle, and the position of any static obstacles (and perhaps other agents). The LV needs to be updated once the new velocity $\vec{v}$ is computed. Fixed and dynamic obstacles will be presented in the LV of each agent, not in a global data structure to be shared by all agents. The size of the individual squares in the LV will vary according to the size of the agent.

9.3.3 Hybrid MER-based Rectabout

In our experiments below, all agents are allowed to move at various speeds. Different speeds require different local views to take into account any other agents in their path, given their speed. The size of agents' LVs and of the squares making up their LVs will determine the size of the rectabout for that agent. This will allow agents

independently to calculate a possible collision and place a virtual rectabout on their paths in case they need to use it to avoid collisions. If one agent is moving at a very high speed (e.g. on the motorway or highway), the agent will need a larger view to react to any hazard and keep a minimum safety distance from other agents. Therefore, LVs and minimal safety distances are scaled by velocity. The larger the LV, the further ahead the agents can plan. If one agent's speed is one grid (agent size $\Phi(A)$ is also one grid) at one move, then setting that agent's LV to grid size two can be guaranteed collision-free. According to Definition 5 and 6, different agents may have various physical sizes and the handling of agents of different sizes is taken into account. Thus, we can write the relationship between minimum local view LV, speed and physical sizes for agents as

$$LV_{min} = \begin{cases} \pi(\|\vec{v}\| + \Phi(A)) & \text{if } \|\vec{v}\| > 0, \\ 0 & \text{if } \|\vec{v}\| = 0. \end{cases} \tag{9.3.1}$$

Equation 9.2.4 is not applicable to multiple agent systems with various speeds. According to Equation 9.3.1, an agent's speed affects that agent's LV and a larger LV affects roundabout location. In hybrid speed multi-agent systems, we can have a simple formula to calculate the minimal safety distance:

$$\Omega(t) = \|\vec{v}\|, \|\vec{v}\| > 0 \tag{9.3.2}$$

and therefore Equation 9.2.4 can be rewritten for agent A_i calculation as

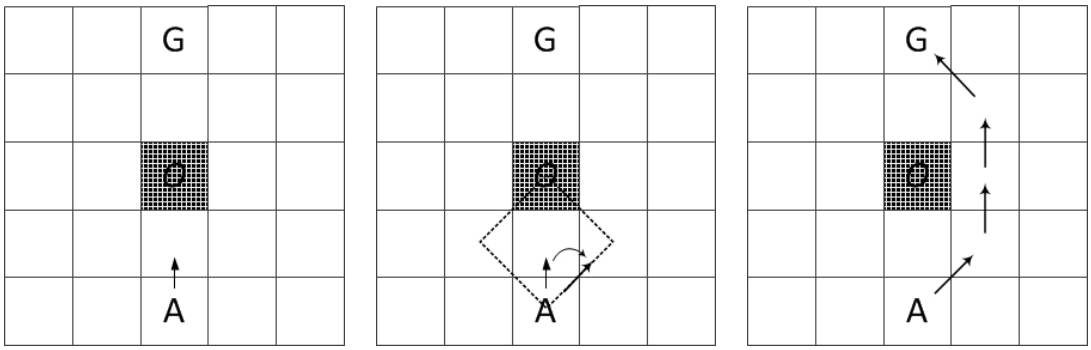
$$\forall t, \forall p_i, p_j, i \neq j, MPD_{ij}(t) > \|\vec{v_i}\|, \tag{9.3.3}$$

which can be applicable to hybrid speed multiple agent systems.

9.3.4 Static Obstacles

We have discussed how agents avoid collisions with each other, but typical multi-agent environments also contain static obstacles. We can follow the same approach as above, with a key difference being that fixed obstacles do not move, so they can be treated as object $\|\vec{v}\| = 0$. We can generally assume that obstacles are modeled as the same size of grid unit due to our simulations being based on a grid environment. Let **O** be a one grid unit static obstacle, and let A be an agent positioned at p_A. Then the virtual rectabout induced by obstacle **O** is defined as (see Figure 9.2(a) and (b)):

$$R^{AO} = MER(p_A, p_O) \tag{9.3.4}$$



(a) (b) (c)

Figure 9.2: A and G represent agent A and agent A's goal, respectively. (a) A configuration of an agent A and a static obstacle **O**. (b) Geometric illustration of how a rectabout is located to resolve collision between the agent and static obstacle using hybrid rectabout. (c) Here the path for the agent is tracked for avoiding the static obstacle using keep right traffic rule.

In case of obstacles, the agent employs the hybrid rectabout to calculate a new velocity to move around such obstacles. This guarantees that there always exists a

valid velocity for the agent that avoids collisions with the fixed obstacle. The direction of motion around obstacles towards the agent's goal can be obtained by the agent using standard path planning techniques, e.g. the A* algorithm [89]. Figure 9.2(c) shows the tracked path, how the agent avoids the static obstacle to reach its goal.

9.4 Experimentation

In this section, we describe the implementation of our approach and report results from our simulation experiments involving multiple autonomous agents.

9.4.1 Implementation Details

We implemented our approach for a set of WowWee Rovio mobile robots using independent sensing and WiFi-based wireless remote control. The WowWee Rovio is a differential-drive mobile robot. It has three individual omni-directional wheels. There are ten various drive and turn speeds in both forward and reverse directions and its shape is a rectangular car-like robot.

All calculations were performed on a 3.2GHz Intel Core i5 system with 4GB of memory running Microsoft Windows XP. However, to ensure that our approach applies when each agent uses its own on-board sensing and mobile laptop for computing, only the WiFi signal sending was carried out centrally. The calculations for each agent were performed in separate and independent processes that did not communicate with each other.

9.4.2 Experimental Results

Using the WowWee Rovio mobile robots, we tested our approach in the following two scenarios.

1. Two robots are deployed on two sides of the field and have to move to their goal positions on the other side using the hybrid rectabout to avoid collision. The video link is `http://youtu.be/nitsN0Sxs9Q`.

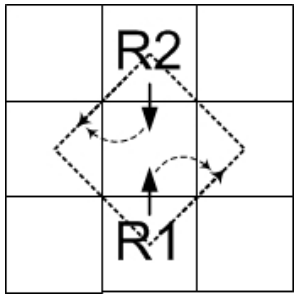


Figure 9.3: Solid arrow line is the intended trajectory. Dotted arrow line is the deconfliction trajectory. The central dotted rectangle is a virtual rectabout enclosing two robots R1 and R2.

2. Four robots are distributed evenly on a square, and their goal is to navigate to the antipodal position on the circle. In doing so, the robots will form a dense crowd in the middle. The video link is `http://youtu.be/lYQY3TZJzwM`.

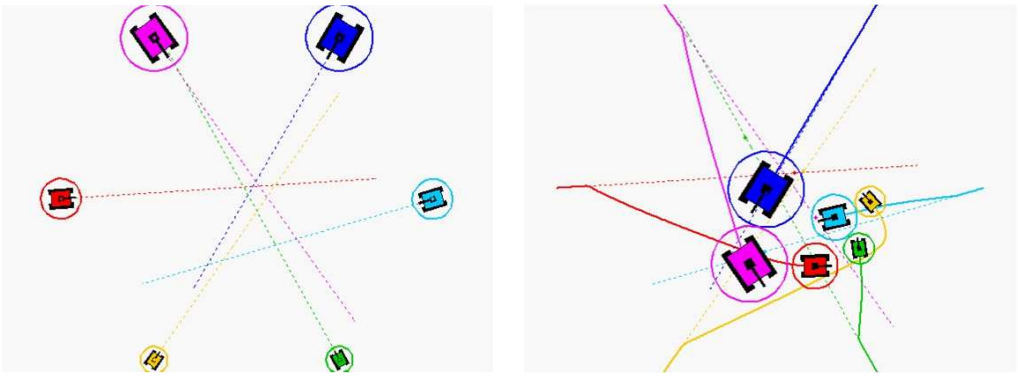
In addition, we tested the heterogeneity and scalability of our approach in the following two simulated scenarios.

1. **Heterogeneity:** The simulation demonstrates a heterogeneous group of five virtual agents navigating from one side to the other, negotiating around each other in the center. For the path computation, each agent employs the A* algorithm [89] to navigate from the initial position to the goal position with

Figure 9.4: Illustration shows the start positions of four robots.

a minimum local view. Each agent is able to detect conflict with any other agent and computes a virtual rectabout based on MER ($MER(p_1, p_2)$). A new velocity is planned along the path of MER and the agents follow a shared 'keep right' traffic rule to resolve the conflict independently. The virtual rectabout is removed after one time-step, after which each agent needs to independently operate the process again, since the information around the agent always changes with every time step. However, agents always attempt to use A* planning path towards their own goal position at every time-step. Figure 9.5 shows snapshots of collision avoidance for six agents while adhering to their chosen paths. The video link is `http://youtu.be/lXHEi0LScXY`.

2. **Scalability:** In order to test the performance of our method we varied the number of agents in a different configuration space (200x200 grid, 300x300 grid and 400x400 grid) to see how our approach scales when the number of agents increases. We performed our experiments on an Intel Core(TM) i5 processor 3.20 GHz with 4GByte of memory. Each scenario was repeated 10 times and



(a) (b)

Figure 9.5: Six hybrid agents (variable size and speed) avoiding collision with each other. (a) soon after starting. (b) after avoiding collisions.

results were averaged. All start and goal positions were generated randomly. All agents have various speeds (1 to 3 grid per move step) and these were randomly assigned for each agent at the initial position. The agents with higher speed require larger local views, leading to increasing computation time in comparison to homogeneous settings. Figure 9.6 shows the total running time for various numbers of agents with various speeds. We note that the total running time of our method scales nearly linearly with the number of heterogeneous agents. Furthermore, the computation time increases as the density of agents increases, as would be expected given that deadlock is more frequent with high density. An agent enters a 'wait' state for one or more moves if it is in deadlock. As long as one agent in a deadlock situation can move, such deadlock is temporary [74].

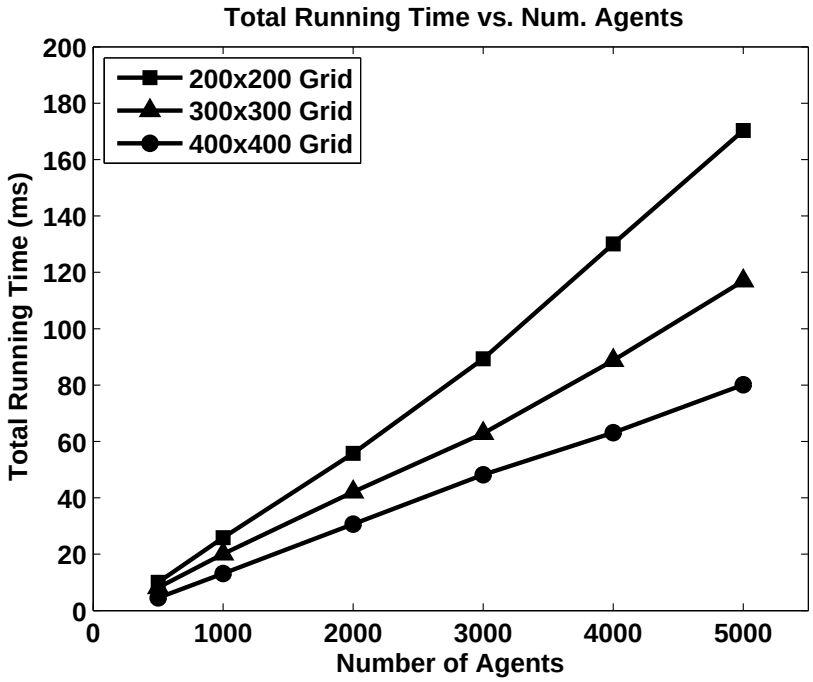


Figure 9.6: The total running can be seen to scale almost linearly with the number of heterogeneous agents.

9.4.3 Comparison

We conducted a number of simulations for comparison against other decentralized approaches. Through these simulations, we tried to evaluate the following two criteria to compare with two decentralized approaches - Satisficing Game Theory (SGT) algorithm by Hill et al. [49, 5] and Distributed Reactive Collision Avoidance (DRCA) algorithm by Lalish and Morgansen [63]: (1) Computation Time: how long the algorithm needs to compute deconfliction between agents in path conflict; and (2) Solution Efficiency: how efficient the solution is for collision avoidance. The scenario is referred to as a choke point because, without deconfliction, all the agents would meet at the center. In order to fairly compare these three algorithms, the hybrid rectabout algorithm has been set up for this simulation such that each agent has a

constant speed (1 grid / move). The agent's size is the same as the grid size, so each agent has information only within the two grid front local view of itself, which is the same as in Hill et al. [49, 5] and Lalish [63].

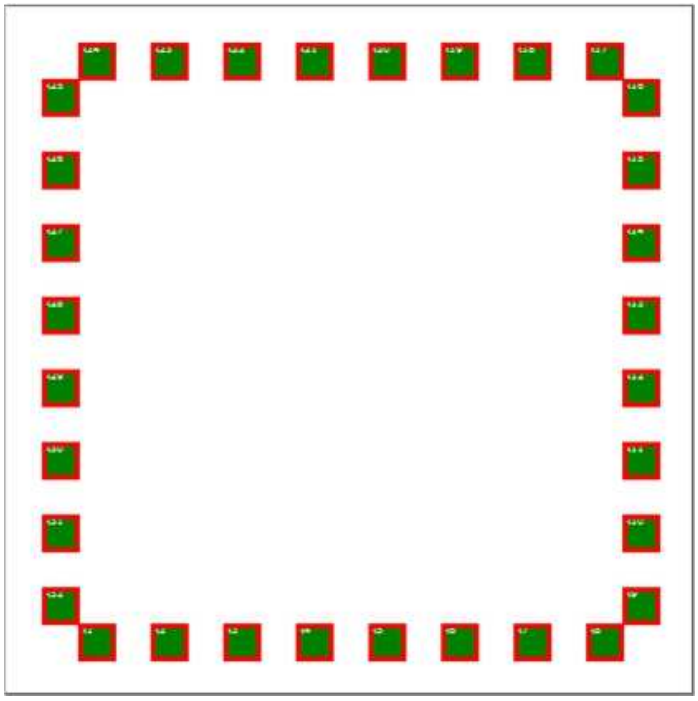


Figure 9.7: Collision avoidance of 32 agents using the hybrid rectabout algorithm where each is attempting to reach a goal on the exact opposite side of the environment (the starting point of the opposite agent).

Computation Time: A simulation of the hybrid rectabout algorithm for deconflicting 32 agents is shown in Figure 9.7, the densest choke pattern demonstrated by Hill et al. [49, 5]. Our hybrid rectabout algorithm consumed 0.1953 milliseconds for the mean running time for deconfliction, compared to 0.1180 seconds in DRCA and 127.3804 seconds in SGT. The reason for the increased time in SGT is that their approach requires communication and priority for deconfliction, unlike our approach.

Solution Efficiency: The efficiency of the maneuver is defined as the average of

the percentage of moving cost (or time delay) in arrival from start position to goal position:

$$Efficiency = \frac{1}{n}\sum_{i=1}^{n}\frac{C_r^i}{C^i},$$

where C_r^i is the reference moving cost for the ith agent (moving straight without considering other agents), and C^i is the actual moving cost taken for the ith agent. The hybrid rectabout algorithm attained an efficiency of 88.9%, compared to 87.6% in DRCA and 85.7% in SGT. However, the DRCA algorithm breaks some of the guarantees of safety for this simulation; in other words, it does not always work for this situation with safety. Meanwhile, the SGT algorithm has collisions occurring in its experiments (recorded 19 out of 32 agents). Importantly, no collisions occurred with our hybrid rectabout algorithm which guaranteed safety for each agent.

We also conducted an experiment with 100 virtual agents moving simultaneously across a circle; the scenario is referred to by [124]. All agents approaching the center of the circle and the agents moving toward the perimeter of the circle have to pass through the center. The timings of this scenario for the hybrid rectabout and three other variations of velocity obstacles (Velocity Obstacle [35], Optimal Reciprocal Collision Avoidance (ORCA) [135], Hybrid Reciprocal Velocity Obstacle [124]) are shown in Table 9.1 These approaches are implemented by an open source library [35, 135, 124].

Figure 9.8 shows the comparison of the timings for this scenario with an increasing number of virtual agents moving across a circle. The timing of the hybrid rectabout requires less time to complete.

Algorithm	Computation Time (ms)
Hybrid Rectabout	**0.77**
Velocity Obstacle	0.81
Optimal Reciprocal Collision Avoidance	0.83
Hybrid Reciprocal Velocity Obstacle	1.24

Table 9.1: Timing of simulations of 100 virtual agents moving simultaneously across a circle using Hybrid Rectabout and three variations of velocity obstacle algorithm.

9.5 Conclusions

Collision avoidance has a long history in both agent-based and robotics research and there exist many approaches, only some of which have been mentioned here. Nearly all previous approaches assume some degree of communication, access to a global map, priority allocation or central coordination. In this chapter, we have introduced a novel and intelligent hybrid rectabout procedure for navigation of multiple robots or autonomous agents using nature-inspired techniques. We take into account the obstacles in the environment as well as uncertainty in position and velocity. We also consider the dynamics and kinematics of the agents, thereby allowing us to implement our approach on WowWee Rovio mobile robots. The kinematic research on minimal predicted distance between two human walkers is applied for the first time to deal with agent collision problems in conjunction with a hybrid rectabout maneuver. We use MPD to detect the possible collisions in agent trajectories. The agents involved in conflict will compute a rectabout and re-plan a new velocity when MPD is below the threshold. This process is repeated until all agents achieve their goals, but each of the agents acts completely independently without central coordination and does not communicate with other agents.

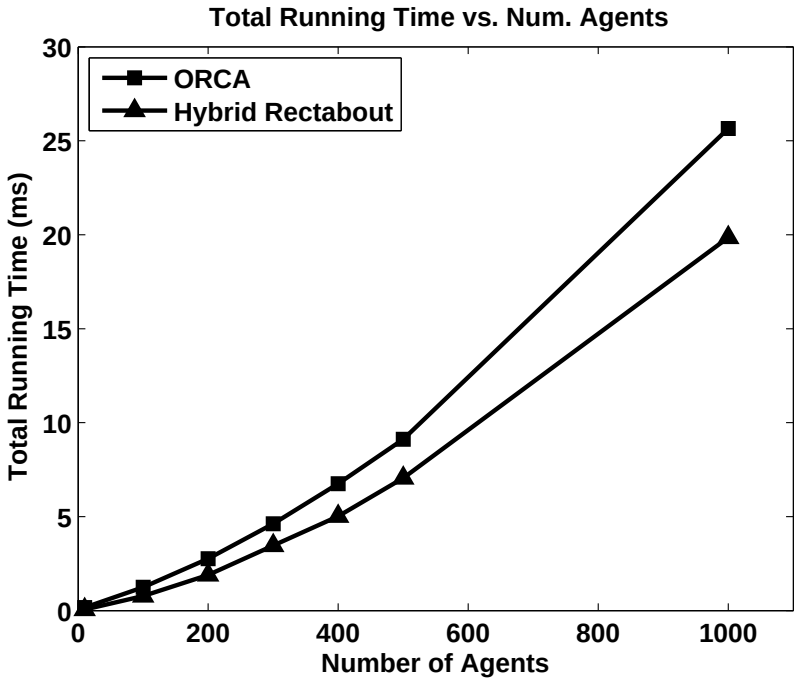


Figure 9.8: Comparison of the timing of simulations of increasing numbers of virtual agents moving simultaneously across a circle of increasing circumference between hybrid rectabout and ORCA.

The hybrid rectabout for mutual avoidance provides a powerful method for a multiple heterogeneous agent avoidance maneuver. At present, most agent search-based algorithms assume all agents have the same physical size (e.g. particles in PSO), travel at the same speed (e.g. ants in ACO) or have the same kinematic constraints (e.g. autonomous robots). Simulations involving such autonomous agents rarely take into account the need to avoid collision, which makes the application of these algorithms to real-world situations problematic. Our findings indicate that a MER-based rectabout procedure can be appended to the search algorithm (e.g. A*) used by agents with little additional cost, resulting in greater applicability to real-world navigation and therefore increased plausibility. We would like to develop a more sophisticated model

of uncertainty that takes into account uncertainty in position and velocity as given by sensors of the agent, and apply it to the hybrid rectabout formulation. The other future direction is to apply our approach for avoiding collisions within swarms or groups of agents with no communication or central coordination. Such collision avoidance is often overlooked or ignored in mobile multi-agent and swarm-based approaches and simulations. Swarm simulations frequently assume that swarm members can fly through each other. The lack of collision detection and avoidance severely limits the application of swarm technology to real-life problems (e.g. nanobots taking antiviral molecules to specific cells in the body, fleets of autonomous cars taking humans safely and reliably to their destinations).

To conclude this part of decentralized design, we raise the question of how natural entities avoid each other, posed at the beginning of this thesis, then we provide modelling on how human pedestrians avoid each other; this is implemented in autonomous multi-agent research through nature-inspired observation for the first time. The previous chapter and current chapter focus on inspiration from nature observation and lead to a new solution on collision avoidance, so that the gap of nature-inspired collision avoidance has been filled through the course of this thesis. Also, we guarantee the nature-inspired collision avoidance strategies work safely and reliably. We have given a good start point on both the macro level and micro level for the motivation of this research. In the next chapter, we will give the conclusion and future directions.

Chapter 10

Conclusion and Future Directions

In this final chapter, we first evaluate the research methodology adopted in this thesis. Secondly, the main achievements of this PhD research are summarized. Finally, the thesis closes with a brief discussion of future directions in four possible areas.

10.1 Evaluation of Research Methodology

In this thesis, we studied nature-inspired observation for collision avoidance as a way for detecting and resolving potential collisions in multi-agent systems. While most existing collision avoidance methods are not intelligent and autonomous, we have shown that a fully decentralized (decoupled) approach - no communication, no global view and no priority can be effective.

In our collision avoidance procedure, MER was taken as a size-flexible roundabout representation method, for which collision avoidance in multi-agent systems was conducted for any type of collisions and any distance between two agents. Moreover, MER was inspired by nature through the use of roundabouts and MPD was inspired

by studies on how human pedestrians detect possible collision. The kinematic research on MPD was applied for the first time to deal with robot collision problems in conjunction with a rectabout maneuver. MER rectabout and MPD both served to resolve collision problems through no communication, no global view and no priority. Therefore, the question "can nature inspiration be used in multi-robot and multi-agent systems?" has been successfully answered.

10.2 Summary of Achievements

This PhD study has shown how nature-inspired techniques (i.e. human behaviour modelling) can be adopted in intelligent, autonomous and decentralized robot and agent systems. Modelling the critical aspect of robots and agents avoiding each other through inspiration from nature is a gap in robot and agent research. We have filled this gap through embedding agent research into robot research by exploring an approach between "collision avoidance" and "nature inspiration". The idea of focusing on the unique problem "collision avoidance" in human walkers appears to be beneficial to the multiple agent / robot modelling problems. It allows identification of how intelligent autonomous entities avoid collision with each other when moving from one point to another and allows further study on robotics research. Every research endeavor starts with the objectives that guide the direction of the research. The ultimate objective of this research has been to develop novel collision avoidance methods and systems for multiple agents and specifically for autonomous heterogeneous agent applications.

10.3 Future Directions

This section presents some promising future directions for the development of the methods and modelling in autonomous multiple mobile agent systems. However, the problems in decentralized multi-agent systems are in principle very challenging and difficult due to the dynamic environments and the lack of efficient methods. Although this study has proposed new algorithms and methods for autonomous agents in collision avoidance and large scale problems, there are limitations and open research problems that need to be investigated and solved in future research.

10.3.1 Limitations

One major restriction on the rectabout method is that the algorithm cannot perform well in an environment that is not large enough to locate a rectabout. This is because the virtual MER-based rectabout lies in the intersecting and conflicting position of two agent paths. If the rectabout cannot be located in-between two conflict agents due to lack of space, the algorithm fails.

A second limitation is that the algorithm does not allow agents to move in formation, because communication between agents is not allowed. It might be possible to employ some swarm techniques, such that PSO algorithm could merge into the rectabout algorithm with no communication. This method would allow formation of agents within the rectabout architecture, but the safety guarantees would have to be re-proven.

A third limitation is that the A* algorithm is used for the path computation from the initial position to the goal position with a minimum local view. The computation does not consider other agents' positions and communication is not allowed. Each

agent is able to detect conflict with the others and computes a virtual rectabout based on MER. However, the optimality for the rectabout with A* is still open. The optimality could be time-optimal, cost-optimal, or velocity optimal.

A fourth limitation is as follows. This thesis focuses on a two dimensional model, because the point of developing and implementing it is for autonomous vehicles. However, we would like to develop a more sophisticated model of three dimensions that takes into account kinematics mechanisms, such as aircrafts being unable to easily stop and go backwards, thus the three dimensional case needs a better deconfliction maneuver, and perhaps this can be accomplished through a hyperplane model, i.e. minimum enclosing ball (MEB) [144, 62, 134, 83]. Badoiu [8] found that the size of the MEB core set is independent of the dimensionality. Based on such a size-flexible characteristic, the MEB can serve as a new 3D model representation in an intelligent collision avoidance approach.

10.3.2 Reality

The whole point of developing these algorithms is to implement them on real systems to improve safety. A big part of the motivation for this research is to integrate the autonomous car [145] (also known as a robotic car, or informally as driverless or self-driving car, an autonomous vehicle capable of fulfilling the human transportation capabilities of a traditional car, e.g. Google driverless car [125]) into the traffic control system, and perhaps also automate more of the existing traffic control system (e.g. becoming traffic-light free). Additionally, this research could be used for collision avoidance regarding autonomous harbour control for ships or teams of mobile agents. These scenarios will become more important as unmanned vehicles are introduced.

Bibliography

[1] Alok Aggarwal, Hiroshi Imai, Naoki Katoh, and Subhash Suri, *Finding k Points With Minimum Diameter And Related Problems*, Journal of Algorithms **12** (1991), no. 1, 38–56.

[2] Rachid Alami, Sara Fleury, Matthieu Herrb, Félix Ingrand, and Frédéric Robert, *Multi-robot Cooperation in the MARTHA Project*, IEEE Robotics & Automation Magazine **5** (1998), no. 1, 36–47.

[3] Brian F. Allen, Nadia Magnenat-Thalmann, and Daniel Thalmann, *Politeness Improves Interactivity in Dense Crowds*, Journal of Visualization and Computer Animation **23** (2012), no. 6, 569–578.

[4] Gianluca Antonelli and Stefano Chiaverini, *Kinematic Control of Platoons of Autonomous Vehicles*, IEEE Transactions on Robotics **22** (2006), no. 6, 1285–1292.

[5] James K. Archibald, Jared C. Hill, N. A. Jepsen, Wynn C. Stirling, and Richard L. Frost, *A Satisficing Approach to Aircraft Conflict Resolution*, IEEE Transactions on Systems, Man, and Cybernetics, Part C **38** (2008), no. 4, 510–521.

[6] Hajime Asama, Koichi Ozaki, Hiroaki Itakura, Akihiro Matsumoto, Yoshiki Ishida, and Isao Endo, *Collision Avoidance Among Multiple Mobile Robots based*

on Rules and Communication, Proceedings of The 1991 IEEE/RSJ International Workshop on Intelligent Robots and Systems' Intelligence for Mechanical Systems (IROS 1991) (Osaka, Japan), 3-5 November 1991, pp. 1215–1220.

[7] Kianoush Azarm and Günther Schmidt, *Conflict-free Motion of Multiple Mobile Robots based on Decentralized Motion Planning and Negotiation*, Proceedings of the 1997 IEEE International Conference on Robotics and Automation (ICRA 1997) (Albuquerque, USA), 20-25 April 1997, pp. 3526–3533.

[8] Mihai Badoiu and Kenneth L. Clarkson, *Optimal Core-sets for Balls*, Computational Geometry **40** (2008), no. 1, 14–22.

[9] Tucker Balch and Ronald C. Arkin, *Behavior-based Formation Control for Multi-robot Teams*, IEEE Transactions on Robotics and Automation **14** (1998), no. 6, 926–939.

[10] Jérôme Barraquand, Bruno Langlois, and Jean-Claude Latombe, *Numerical Potential Field Techniques for Robot Path Planning*, IEEE Transactions on Systems, Man, and Cybernetics **22** (1992), no. 2, 224–241.

[11] Jérôme Barraquand and Jean-Claude Latombe, *Robot Motion Planning: A Distributed Representation Approach*, The International Journal of Robotics Research **10** (1991), no. 6, 628–649.

[12] Calin Belta and Vijay R. Kumar, *Abstraction and Control for Groups of Robots*, IEEE Transactions on Robotics **20** (2004), no. 5, 865–875.

[13] Maren Bennewitz, Wolfram Burgard, and Sebastian Thrun, *Optimizing Schedules for Prioritized Path Planning of Multi-Robot Systems*, Proceedings of the 2001 IEEE International Conference on Robotics and Automation (ICRA 2001) (Seoul, Korea), 21-26 May 2001, pp. 271–276.

[14] ———, *Finding And Optimizing Solvable Priority Schemes For Decoupled Path Planning Techniques For Teams Of Mobile Robots*, Robotics and Autonomous Systems **41** (2002), no. 2-3, 89–99.

[15] Alan H. Bond and Les Gasser, *An Analysis of Problems and Research in DAI*, Distributed Artificial Intelligence (Alan H. Bond and Les Gasser, eds.), Morgan Kaufmann Publishers, San Mateo, CA, 1988, pp. 3–35.

[16] Barry Brumitt and Anthony Stentz, *Dynamic Mission Planning for Multiple Mobile Robots*, Proceedings of the 1996 IEEE International Conference on Robotics and Automation (ICRA 1996) (Minneapolis, USA), 22-28 April 1996, pp. 2396–2401.

[17] ———, *GRAMMPS: A Generalized Mission Planner for Multiple Mobile Robots in Unstructured Environments*, Proceedings of the 1998 IEEE International Conference on Robotics and Automation (ICRA 1998) (Leuven, Belgium), 16-20 May 1998, pp. 1564–1571.

[18] Stephen J. Buckley, *Fast Motion Planning for Multiple Moving Robots*, Proceedings of the 1989 IEEE International Conference on Robotics and Automation (ICRA 1989) (Scottsdale, USA), 14-19 May 1989, pp. 322–326.

[19] Schubert R. Carvalho, Ronan Boulic, Creto Augusto Vidal, and Daniel Thalmann, *Latent Motion Spaces for Full-body Motion Editing*, The Visual Computer **29** (2013), no. 3, 171–188.

[20] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A. Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, Cambridge, MA, June 2005.

[21] Li Chun, Zhiqiang Zheng, and Wensen Chang, *A Decentralized Approach to the Conflict-Free Motion Planning for Multiple Mobile Robots*, Proceedings of the 1999 IEEE International Conference on Robotics and Automation (ICRA 1999) (Detroit, USA), 10-15 May 1999, pp. 1544–1549.

[22] Christopher M. Clark, Stephen M. Rock, and Jean-Claude Latombe, *Motion Planning for Multiple Mobile Robots using Dynamic Networks*, Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA 2003) (Taipei, Taiwan), 14-19 September 2003, pp. 4222–4227.

[23] Sandip Das, Partha P. Goswami, and Subhas C. Nandy, *Smallest k-point Enclosing Rectangle And Square Of Arbitrary Orientation*, Information Processing Letters **94** (2005), no. 6, 259–266.

[24] Minati De, Anil Maheshwari, Subhas C. Nandy, and Michiel H. M. Smid, *An In-Place Min-Max Priority Search Tree*, Computational Geometry **46** (2013), no. 3, 310–327.

[25] Dimos V. Dimarogonas, Savvas G. Loizou, Kostas J. Kyriakopoulos, and Michael M. Zavlanos, *A Feedback Stabilization and Collision Avoidance Scheme for Multiple Independent Non-point Agents*, Automatica **42** (2006), no. 2, 229–243.

[26] Marco Dorigo, Marco Antonio Montes de Oca, and Andries Petrus Engelbrecht, *Particle Swarm Optimization*, Scholarpedia **3** (2008), no. 11, 1486.

[27] Marco Dorigo and Luca Maria Gambardella, *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem*, IEEE Transactions on Evolutionary Computation **1** (1997), no. 1, 53–66.

[28] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni, *Ant System: Optimization by a Colony of Cooperating Agents*, IEEE Transactions on Systems, Man, and Cybernetics, Part B **26** (1996), no. 1, 29–41.

[29] Zvi Drezner and Horst W. Hamacher, *Facility Location: Applications and Theory*, Springer, Berlin, 2002.

[30] Magnus Egerstedt and Xiaoming Hu, *Formation Constrained Multi-Agent Control*, IEEE Transactions on Robotics **17** (2001), no. 6, 947–951.

[31] David Eppstein and Jeff Erickson, *Iterated Nearest Neighbors And Finding Minimal Polytopes*, Discrete & Computational Geometry **11** (1994), no. 1, 321–350.

[32] Michael Erdmann and Tomás Lozano-Pérez, *On Multiple Moving Objects*, Algorithmica **2** (1987), no. 1-4, 477–521.

[33] J. Alexander Fax and Richard M. Murray, *Information Flow and Cooperative Control of Vehicle Formations*, IEEE Transactions on Automatic Control **49** (2004), no. 9, 1465–1476.

[34] Carlo Ferrari, Enrico Pagello, Jun Ota, and Tamio Arai, *Multirobot Motion Coordination in Space and Time*, Robotics and Autonomous Systems **25** (1998), no. 3-4, 219–229.

[35] Paolo Fiorini and Zvi Shiller, *Motion Planning in Dynamic Environments Using Velocity Obstacles*, International Journal of Robotic Research **17** (1998), no. 7, 760–772.

[36] Atsushi Fujimori, Masato Teramoto, Peter N. Nikiforuk, and Madan M. Gupta, *Cooperative Collision Avoidance between Multiple Mobile Robots*, Journal of Robotic Systems **17** (2000), no. 7, 347–363.

[37] Veysel Gazi, *Swarm Aggregations using Artificial Potentials and Sliding-Mode Control*, IEEE Transactions on Robotics **21** (2005), no. 6, 1208–1214.

[38] Pascal Glardon, Ronan Boulic, and Daniel Thalmann, *Dynamic Obstacle Avoidance for Real-time Character Animation*, The Visual Computer **22** (2006), no. 6, 399–414.

[39] Siome Goldenstein, Menelaos I. Karavelas, Dimitris N. Metaxas, Leonidas J. Guibas, Eric Aaron, and Ambarish Goswami, *Scalable Nonlinear Dynamical Systems for Agent Steering and Crowd Simulation*, Computers & Graphics **25** (2001), no. 6, 983–998.

[40] Alfred Scharff Goldhaber and Michael Martin Nieto, *Photon and Graviton Mass Limits*, Reviews of Modern Physics **82** (2010), no. 1, 939–979.

[41] Norman C. Griswold and J. Eem, *Control for Mobile Robots in the Presence of Moving Objects*, IEEE Transactions on Robotics and Automation **6** (1990), no. 2, 263–268.

[42] David D. Grossman, *Traffic Control of Multiple Robot Vehicles*, IEEE Transactions on Robotics and Automation **4** (1988), no. 5, 491–497.

[43] Dongbing Gu and Erfu Yang, *Fuzzy Policy Reinforcement Learning in Cooperative Multi-robot Systems*, Journal of Intelligent and Robotic Systems **48** (2007), no. 1, 7–22.

[44] Yi Guo and Lynne E. Parker, *A Distributed and Optimal Motion Planning Approach for Multiple Mobile Robots*, Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA 2002) (Washington, USA), 11-15 May 2002, pp. 2612–2619.

[45] Yi Guo, Lynne E. Parker, and Raj Madhavan, *Towards Collaborative Robots for Infrastructure Security Applications*, Proceedings of the International Symposium on Collaborative Technologies and Systems (CTS 2004) (San Diego, USA), 18-23 January 2004, pp. 235–240.

[46] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael, *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*, IEEE Transactions on Systems Science and Cybernetics **4** (1968), no. 2, 100–107.

[47] Barbara Hayes-Roth, *An Architecture for Adaptive Intelligent Systems*, Artificial Intelligence **72** (1995), no. 1-2, 329–365.

[48] Christopher J. Hazard, Peter R. Wurman, and Raffaello D'Andrea, *Alphabet Soup: A Testbed for Studying Resource Allocation in Multi-vehicle Systems*, Proceedings of the AAAI Workshop on Auction Mechanisms for Robot Coordination (AMRC 2006) (Boston, USA), 17 July 2006, pp. 23–30.

[49] Jared Hill, James Archibald, Wynn Stirling, and Richard Frost, *A Multi-Agent System Architecture for Distributed Air Traffic Control*, Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit (AIAA 2005) (San Francisco, California, USA), 15-18 August 2005, pp. 1936–1946.

[50] John E. Hopcroft, Jacob T. Schwartz, and Micha Sharir, *On the Complexity of Motion Planning for Multiple Independent Objects; PSPACE-Hardness of the "Warehouseman's Problem"*, The International Journal of Robotics Research **3** (1984), no. 4, 76–88.

[51] Yong K. Hwang and Narendra Ahuja, *Gross Motion Planning - A Survey*, ACM Computing Surveys **24** (1992), no. 3, 219–291.

[52] Ali Jadbabaie, Jie Lin, and A. Stephen Morse, *Coordination of Groups of Mobile Autonomous Agents using Nearest Neighbor Rules*, IEEE Transactions on Automatic Control **48** (2003), no. 6, 988–1001.

[53] Markus Jagër and Bernhard Nebel, *Decentralized Collision Avoidance, Deadlock Detection, and Deadlock Resolution for Multiple Mobile Robots*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001) (Maui, USA), 29 October - 3 November 2001, pp. 1213–1219.

[54] M. Renee Jansen and Nathan R. Sturtevant, *Direction Maps for Cooperative Pathfinding*, Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2008) (Stanford, USA), 22-24 October 2008, pp. 185–190.

[55] James S. Jennings, Greg Whelan, and William F. Evans, *Cooperative Search and Rescue with a Team of Mobile Robots*, Proceedings of the 1997 IEEE International Conference on Robotics and Automation (ICRA 1997) (Monterey, USA), 7-9 July 1997, pp. 193–200.

[56] Yaochu Jin and Yan Meng, *Morphogenetic Robotics: An Emerging New Field in Developmental Robotics*, IEEE Transactions on Systems, Man, and Cybernetics, Part C **41** (2011), no. 2, 145–160.

[57] Shin Kato, Sakae Nishiyama, and Jun'ichi Takeno, *Coordinating Mobile Robots by Applying Traffic Rules*, Proceedings of The 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 1992) (Raleigh, USA), 7-10 July 1992, pp. 1535–1541.

[58] Lydia E. Kavraki, Mihail N. Kolountzakis, and Jean-Claude Latombe, *Analysis of Probabilistic Roadmaps for Path Planning*, IEEE Transactions on Robotics and Automation **14** (1998), no. 1, 166–171.

[59] Lydia E. Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark H. Overmars, *Probabilistic Roadmaps for Path Planning in High-dimensional Configuration Spaces*, IEEE Transactions on Robotics and Automation **12** (1996), no. 4, 566–580.

[60] Oussama Khatib, *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots*, The International Journal of Robotics Research **5** (1986), no. 1, 90–98.

[61] Ross A. Knepper and Daniela Rus, *Pedestrian-Inspired Sampling-based Multi-robot Collision Avoidance*, Proceedings of The 21st IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2012) (Paris, France), 9-13 September 2012, pp. 94–100.

[62] Piyush Kumar, Joseph S. B. Mitchell, and E. Alper Yildirim, *Approximate Minimum Enclosing Balls in High Dimensions using Core-sets*, Journal of Experimental Algorithmics (JEA) **8** (2003), no. 1.1.

[63] Emmett Lalish and Kristi A. Morgansen, *Distributed Reactive Collision Avoidance*, Autonomous Robots **32** (2012), no. 3, 207–226.

[64] Fabrice Lamarche and Stéphane Donikian, *Crowd of Virtual Humans: a New Approach for Real Time Navigation in Complex and Structured Environments*, Computer Graphics Forum **23** (2004), no. 3, 509–518.

[65] Jean-Claude Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, 1991.

[66] Steven M. LaValle, *Planning Algorithms*, Cambridge University Press, New York, USA, 2006.

[67] Steven M. LaValle and Seth Hutchinson, *Optimal Motion Planning for Multiple Robots Having Independent Goals*, IEEE Transactions on Robotics and Automation **14** (1998), no. 6, 912–925.

[68] B. H. Lee and C. S. George Lee, *Collision-Free Motion Planning of Two Robots*, IEEE Transactions on Systems, Man, and Cybernetics **17** (1987), no. 1, 21–32.

[69] Byoung-Ju Lee, Sung-Oh Lee, and Gwi-Tae Park, *Trajectory Generation and Motion Tracking Control for the Robot Soccer Game*, Proceedings of The 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 1999) (Kyongju, Korea), 17-21 October 1999, pp. 1149–1154.

[70] Jihong Lee and Zeungnam Bien, *Collision-free Trajectory Control for Multiple Robots based on Neural Optimization Network*, Robotica **8** (1990), no. 3, 185–194.

[71] Hui Liang, Junsong Yuan, Daniel Thalmann, and Zhengyou Zhang, *Model-based Hand Pose Estimation via Spatial-temporal Hand Parsing and 3D Fingertip Localization*, The Visual Computer **29** (2013), no. 6-8, 837–848.

[72] Chi-Fang Lin and Wen-Hsiang Tsai, *Motion Planning for Multiple Robots with Multi-mode Operations via Disjunctive Graphs*, Robotica **9** (1991), no. 4, 393–408.

[73] Fan Liu and Ajit Narayanan, *Real Time Replanning based on A* for Collision Avoidance in Multi-Robot Systems*, Proceedings of the 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2011) (Incheon, South Korea), 23-26 November 2011, pp. 473–479.

[74] ———, *A Human-Inspired Collision Avoidance Method for Multi-robot and Mobile Autonomous Robots*, Proceedings of the 16th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA 2013) (Dunedin, New Zealand), 1-6 December 2013, pp. 181–196.

[75] ———, *Roundabout Collision Avoidance for Multiple Robots based on Minimum Enclosing Rectangle (Demonstration)*, Proceedings of the International

conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2013) (Saint Paul, USA), 6-10 May 2013, pp. 1375–1376.

[76] Fan Liu, Ajit Narayanan, and Quan Bai, *Effective Methods For Generating Collision Free Paths For Multiple Robots Based On Collision Type (Demonstration)*, Proceedings of the Eleventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012) (Valencia, Spain), 22-24 June 2012, pp. 1459–1460.

[77] Brian Logan and Natasha Alechina, *A* with Bounded Costs*, Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference (AAAI 1998, IAAI 1998) (Madison, USA), 26-30 July 1998, pp. 444–449.

[78] V.J. Lumelsky and K.R. Harinarayan, *Decentralized Motion Planning for Multiple Mobile Robots: The Cocktail Party Model*, Autonomous Robots **4** (1997), no. 1, 121–135.

[79] Priya Ranjan Sinha Mahapatra, Arindam Karmakar, Sandip Das, and Partha P. Goswami, *k-Enclosing Axis-Parallel Square*, Proceedings of the 2011 International Conference on Computational Science and Its Applications (ICCSA 2011) (Santander, Spain), 20-23 June 2011, pp. 84–93.

[80] Joshua A. Marshall, Mireille E. Broucke, and Bruce A. Francis, *Formations of Vehicles in Cyclic Pursuit*, IEEE Transactions on Automatic Control **49** (2004), no. 11, 1963–1974.

[81] W. S. Massey, *Cross Products of Vectors in Higher Dimensional Euclidean Spaces*, The American Mathematical Monthly **90** (1983), no. 10, 697–701.

[82] Maja J Matarić, *Designing Emergent Behaviors: From Local Interactions to Collective Intelligence*, Proceedings of the Second International Conference on

Simulation of Adaptive Behavior (SAB 1992) (Cambridge, USA), MIT Press, 1992, pp. 432–441.

[83] Nimrod Megiddo, *Linear-Time Algorithms for Linear Programming in R^3 and Related Problems*, SIAM Journal on Computing **12** (1983), no. 4, 759–776.

[84] Gavin S. P. Miller, *Snake Robots for Search and Rescue*, Neurotechnology for Biomimetic Robots (Joseph Ayers, Joel L. Davis, and Alan Rudolph, eds.), MIT Press, Cambridge, MA, 2002, pp. 271–284.

[85] Sergio Monteiro and Estela Bicho, *A Dynamical Systems Approach to Behavior-Based Formation Control*, Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA 2002) (Washington, USA), 11-15 May 2002, pp. 2606–2611.

[86] Anastasios I. Mourikis and Stergios I. Roumeliotis, *Optimal Sensor Scheduling for Resource-constrained Localization of Mobile Robot Formations*, IEEE Transactions on Robotics **22** (2006), no. 5, 917–931.

[87] ———, *Performance Analysis of Multirobot Cooperative Localization*, IEEE Transactions on Robotics **22** (2006), no. 4, 666–681.

[88] Subhas C. Nandy and Bhargab B. Bhattacharya, *A Unified Algorithm for Finding Maximum and Minimum Object Enclosing Rectangles and Cuboids*, Computers & Mathematics with Applications **29** (1995), no. 8, 45–61.

[89] Nils J. Nilsson, *Principles of Artificial Intelligence*, Springer, Berlin, New York, 1982.

[90] Patrick A. O'Donnell and Tomás Lozano-Pérez, *Deadlock-Free and Collision-Free Coordination of Two Robot Manipulators*, Proceedings of the 1989 IEEE International Conference on Robotics and Automation (ICRA 1989) (Scottsdale, USA), 14-19 May 1989, pp. 484–489.

[91] Anne-Hélène Olivier, Antoine Marin, Armel Grétual, Alain Berthoz, and Julien Pettré, *Collision avoidance between two walkers: Role-dependent strategies*, Gait & Posture **38** (2013), no. 4, 751–756.

[92] Anne-Hélène Olivier, Antoine Marin, Armel Grétual, and Julien Pettré, *Minimal Predicted Distance: A Common Metric For Collision Avoidance During Pairwise Interactions Between Walkers*, Gait & Posture **36** (2012), no. 3, 399–404.

[93] ———, *Minimal Predicted Distance: A Kinematic Cue To Investigate Collision Avoidance Between Walkers*, Computer Methods in Biomechanics and Biomedical Engineering **15** (2012), no. 1, 240–242.

[94] Lucia Pallottino, Vincenzo Giovanni Scordio, Antonio Bicchi, and Emilio Frazzoli, *Decentralized Cooperative Policy for Conflict Resolution in Multivehicle Systems*, IEEE Transactions on Robotics **23** (2007), no. 6, 1170–1183.

[95] Tai-Jee Pan and R.C. Luo, *Motion Planning for Mobile Robots in a Dynamic Environment with Moving Obstacles*, Proceedings of the 1990 IEEE International Conference on Robotics and Automation (ICRA 1990) (Cincinnati, USA), 13-18 May 1990, pp. 578–583.

[96] Shaoning Pang, Fan Liu, Youki Kadobayashi, Tao Ban, and Daisuke Inoue, *Training Minimum Enclosing Balls for Cross Tasks Knowledge Transfer*, Proceedings of the 19th International Conference on Neural Information Processing (ICONIP 2012) (Doha, Qatar), 12-15 November 2012, pp. 375–382.

[97] Lynne E. Parker, *ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation*, IEEE Transactions on Robotics and Automation **14** (1998), no. 2, 220–240.

[98] ———, *Multiple Mobile Robot Systems*, Springer Handbook of Robotics (Bruno Siciliano and Oussama Khatib, eds.), Springer, 2008, pp. 921–941.

[99] ———, *Path Planning and Motion Coordination in Multiple Mobile Robot Teams*, Encyclopedia of Complexity and Systems Science (Robert A. Meyers, ed.), Springer, 2009, pp. 5783–5800.

[100] Lynne E. Parker and John V. Draper, *Robotics Applications in Maintenance and Repair*, Handbook of Industrial Robotics, 2^{nd} Edition (Shimon Nof, ed.), Wiley Publishers, 1999, pp. 1023–1036.

[101] David Parsons and John Canny, *A Motion Planner for Multiple Mobile Robots*, Proceedings of the 1990 IEEE International Conference on Robotics and Automation (ICRA 1990) (Cincinnati, USA), 13-18 May 1990, pp. 8–13.

[102] Simon Parsons and Michael Wooldridge, *Game Theory and Decision Theory in Multi-Agent Systems*, Autonomous Agents and Multi-Agent Systems **5** (2002), no. 3, 243–254.

[103] Mike Peasgood, Christopher M. Clark, and John McPhee, *A Complete and Scalable Strategy for Coordinating Multiple Robots Within Roadmaps*, IEEE Transactions on Robotics **24** (2008), no. 2, 283–292.

[104] Michal Pechoucek and David Sislák, *Agent-Based Approach to Free-Flight Planning, Control, and Simulation*, IEEE Intelligent Systems **24** (2009), no. 1, 14–17.

[105] Julien Pettré, Helena Grillon, and Daniel Thalman, *Crowds of Moving Objects: Navigation Planning and Simulation*, Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA 2007) (Roma, Italy), 10-14 April 2007, pp. 3062–3067.

[106] Thomas Pilarski, Michael Happold, Henning Pangels, Mark Ollis, Kerien Fitzpatrick, and Anthony (Tony) Stentz, *The Demeter System for Automated Harvesting*, Proceedings of the 8th International Topical Meeting on Robotics and Remote Systems (USA), April 1999.

[107] André Platzer and Edmund M. Clarke, *Formal Verification Of Curved Flight Collision Avoidance Maneuvers: A Case Study*, Proceedings of Formal Methods, Second World Congress (FM 2009) (Eindhoven, The Netherlands), 2-6 November 2009, pp. 547–562.

[108] Samer Qutub, Rachid Alami, and Félix Ingrand, *How to Solve Deadlock Situations within the Plan-Merging Paradigm for Multi-robot Cooperation*, Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 1997) (Grenoble, France), 7-11 September 1997, pp. 1610–1615.

[109] Craig W. Reynolds, *Flocks, Herds and Schools: A Distributed Behavioral Model*, Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1987) (New York, USA) (Maureen C. Stone, ed.), ACM Press, 1987, pp. 25–34.

[110] M. Rude, *Collision Avoidance by Using Space-Time Representations of Motion Processes*, Autonomous Robots **4** (1997), no. 1, 101–119.

[111] Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, third edition, Upper Saddle River, New Jersey, 2009.

[112] Malcolm Ryan, *Graph Decomposition for Efficient Multi-robot Path Planning*, Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007) (Hyderabad, India), 6-12 January 2007, pp. 2003–2008.

[113] Jacob T. Schwartz and Micha Sharir, *On the Piano Movers' Problem: III. Coordinating the Motion of Several Independent Bodies: The Special Case of*

Circular Bodies Moving Amidst Polygonal Barriers, The International Journal of Robotics Research **2** (1983), no. 3, 46–75.

[114] ———, *A Survey of Motion Planning and Related Geometric Algorithms*, Artificial Intelligence **37** (1988), no. 1-3, 157–169.

[115] Michael Segal and Klara Kedem, *Enclosing k Points In The Smallest Axis Parallel Rectangle*, Information Processing Letters **65** (1998), no. 2, 95–99.

[116] Gary Shaffer and Anthony Stentz, *A Robotic System for Underground Coal Mining*, Proceedings of the 1992 IEEE International Conference on Robotics and Automation (ICRA 1992) (Nice, France), 12-14 May 1992, pp. 633–638.

[117] Micha Sharir, *Algorithmic Motion Planning in Robotics*, IEEE Computer **22** (1989), no. 3, 9–20.

[118] Guni Sharon, Roni Stern, Ariel Felner, and Nathan Sturtevant, *Conflict-Based Search For Optimal Multi-Agent Path Finding*, Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2012) (Toronto, Canada), 22-26 July 2012, pp. 563–569.

[119] Li-Yen Shue and Reza Zamani, *An Admissible Heuristic Search Algorithm*, Proceedings of the 7th International Symposium, Methodologies for Intelligent Systems (ISMIS 1993) (Trondheim, Norway), 15-18 June 1993, pp. 69–75.

[120] David Silver, *Cooperative Pathfinding*, Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2005) (Marina del Rey, USA), 1-5 June 2005, pp. 117–122.

[121] Reid Simmons, Sanjiv Singh, David Hershberger, Josue Ramos, and Trey Smith, *First Results in the Coordination of Heterogeneous Robots for Large-Scale Assembly*, Proceedings of the Seventh International Symposium on Experimental Robotics (ISER 2000) (Honolulu, USA), Springer-Verlag, 10-13 December 2000.

[122] David Šišlák, Jiří Samek, and Michal Pěchouček, *Decentralized Algorithms for Collision Avoidance in Airspace*, Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008) (Estoril, Portugal), vol. 2, 12-16 May 2008, pp. 543–550.

[123] Igor Škrjanc and Gregor Klančar, *Optimal Cooperative Collision Avoidance Between Multiple Robots Based On Bernstein-Bézier Curves*, Robotics and Autonomous Systems **58** (2010), no. 1, 1–9.

[124] Jamie Snape, Jur van den Berg, Stephen J. Guy, and Dinesh Manocha, *The Hybrid Reciprocal Velocity Obstacle*, IEEE Transactions on Robotics **27** (2011), no. 4, 696–706.

[125] IEEE Spectrum, *How Google's Self-Driving Car Works*, October 2011.

[126] Trevor Scott Standley, *Finding Optimal Solutions to Cooperative Pathfinding Problems*, Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010) (Atlanta, USA), 11-15 July 2010, pp. 173–178.

[127] Peter Stone and Manuela M. Veloso, *Multiagent Systems: A Survey from a Machine Learning Perspective*, Autonomous Robots **8** (2000), no. 3, 345–383.

[128] Ashley W. Stroupe, Avi Okon, Matthew L. Robinson, Terry Huntsberger, Hrand Aghazarian, and Eric T. Baumgartner, *Sustainable Cooperative Robotic Technologies for Human and Robotic Outpost Infrastructure Construction and Maintenance*, Autonomous Robots **20** (2006), no. 2, 113–123.

[129] Petr Svestka and Mark H. Overmars, *Coordinated Path Planning for Multiple Robots*, Robotics and Autonomous Systems **23** (1998), no. 3, 125–152.

[130] Paulo Tabuada, George J. Pappas, and Pedro U. Lima, *Motion Feasibility of Multi-agent Formations*, IEEE Transactions on Robotics **21** (2005), no. 3, 387–392.

[131] Chuck Thorpe, Todd Jochem, and Dean Pomerleau, *The 1997 Automated Highway Free Agent Demonstration*, Proceedings of the 1997 IEEE Conference on Intelligent Transportation System (ITSC 1997) (Boston, USA), 9-12 November 1997, pp. 496–501.

[132] Chad M. Topaz and Andrea L. Bertozzi, *Swarming Patterns in a Two-Dimensional Kinematic Model for Biological Groups*, SIAM Journal of Applied Mathematics **65** (2004), no. 1, 152–174.

[133] Adrien Treuille, Seth Cooper, and Zoran Popovic, *Continuum Crowds*, ACM Transactions on Graphics (TOG) **25** (2006), no. 3, 1160–1168.

[134] Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung, *Core Vector Machines: Fast SVM Training on Very Large Data Sets*, Journal of Machine Learning Research **6** (2005), 363–392.

[135] Jur van den Berg, Stephen J. Guy, Ming C. Lin, and Dinesh Manocha, *Reciprocal n-Body Collision Avoidance*, Proceedings of The 14th International Symposium of Robotics Research (ISRR 2009) (Lucerne, Switzerland), 31 August-3 September 2009, pp. 3–19.

[136] Jur van den Berg, Ming Lin, and Dinesh Manocha, *Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation*, Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA 2008) (Pasadena, USA), 19-23 May 2008, pp. 1928–1935.

[137] Jur van den Berg, Jack Snoeyink, Ming Lin, and Dinesh Manocha, *Centralized Path Planning For Multiple Robots: Optimal Decoupling Into Sequential Plans*, Proceedings of Robotics: Science and Systems (RSS 2009) (Seattle, USA), 28 June - 1 July 2009.

[138] Wouter van Toll, Atlas F. Cook IV, and Roland Geraerts, *Navigation Meshes for Realistic Multi-Layered Environments*, Proceedings of The 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011) (San Francisco, USA), 25-30 September 2011, pp. 3526–3532.

[139] Jing Wang and Gerardo Beni, *Distributed Computing Problems in Cellular Robotic Systems*, Proceedings of The 1990 IEEE/RSJ International Workshop on Intelligent Robots and Systems' Towards a New Frontier of Applications (IROS 1990) (Ibaraki, Japan), 3-6 July 1990, pp. 819–826.

[140] P. K. C. Wang, *Interaction Dynamics of Multiple Autonomous Mobile Robots in Bounded Spatial Domains*, International Journal of Control **50** (1989), no. 6, 2109–2124.

[141] ———, *Interaction Dynamics of Multiple Mobile Robots with Simple Navigation Strategies*, Journal of Robotic Systems **6** (1989), no. 1, 77–101.

[142] Yanbin Wang, Rohit Dubey, Nadia Magnenat-Thalmann, and Daniel Thalmann, *An Immersive Multi-agent System for Interactive Applications*, The Visual Computer **29** (2013), no. 5, 323–332.

[143] Charles W. Warren, *Multiple Robot Path Coordination Using Artificial Potential Fields*, Proceedings of the 1990 IEEE International Conference on Robotics and Automation (ICRA 1990) (Cincinnati, USA), 13-18 May 1990, pp. 500–505.

[144] Emo Welzl, *Smallest enclosing disks (balls and ellipsoids*, New Results and New Trends in Computer Science (Graz, Austria) (Hermann Maurer, ed.), Lecture Notes in Computer Science, vol. 555, Springer-Verlag, 20-21 June 1991, pp. 359–370.

[145] Wikipedia, *Autonomous Car*, May 2013.

[146] ———, *Intelligence*, February 2014.

[147] ———, *Natural computing*, March 2014.

[148] ———, *Robot*, February 2014.

[149] Barbara Yersin, Jonathan Maïm, Fiorenzo Morini, and Daniel Thalmann, *Real-time Crowd Motion Planning*, The Visual Computer **24** (2008), no. 10, 859–870.

[150] Mohan Yogeswaran and S. G. Ponnambalam, *An Extensive Review of Research in Swarm Robotics*, Proceedings of the World Congress on Nature & Biologically Inspired Computing (NaBIC 2009) (Coimbatore, India), 9-11 December 2009, pp. 140–145.

[151] Shin'ichi Yuta and Suparerk Premvuti, *Coordinating Autonomous and Centralized Decision Making to Achieve Cooperative Behaviors between Multiple Mobile Robots*, Proceedings of The 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 1992) (Raleigh, USA), 7-10 July 1992, pp. 1566–1574.