Mälardalen University Doctoral Thesis
No.352

Toward Dependable Multiple Path Planning for Autonomous Robots with Obstacle Avoidance and Congestion Control

Lan Anh Trinh

January, 2022

MÄLARDALEN UNIVERSITY

School of Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden

ISSN 1651-4238
ISBN 978-91-7485-541-8
Printed by E-Print AB, Stockholm, Sweden

To my parents, my husband Thang Nguyen and our lovely kids Khang (Sam) and Lam Anh (Alisa).

Acknowledgments

As time flies, it is being close to the end of my PhD study. Through the PhD life with a lot of pains but also much of happiness, I have gained a lot from the way I have been: research vision, knowledge, practical experiences, patience as well as friendships. The most important word I want to say at this moment is THANK YOU.

I would like to express my thanks to many persons who have directly and indirectly helped me on the path leading to this dissertation. First, I especially express my thanks to my supervisors, Prof. Dr. Mikael Ekström and Dr. Baran Cürüklü, to give me a great chance to experience the most wonderful PhD life at Mälardalen University. I would like to express my sincere gratitude for their valuable supports and advices in both research and life. Thanks to their experience and conversations, I have become more and more open-minded to solve my difficulties. Thanks to their encouragements I did and do overcome all challenges to go through till the end of this PhD study.

Second, I would like to extend my thanks to all MDH gangs, PhD students at Mälardalen University, who are very good, friendly, and enthusiastic. Thanks to the relax moments in every lunches, I understand more about friendships and the culture in alternative countries. Many thanks to all of my friends in Robotics Group, Fredrik, Carl, Henrik, Afshin and many others, who have been always very nice and friendly teammates. I would like to especially thank the group of master students, who have helped me a lots to bring my theoretical research into realistic implementation. And, of course, I don't know how

much thankful to two my office-mates Gita and Branko, for helping me in both researches and life, and for all the happy moments and funny stories so that our office is always full of laughing.

Many next thanks to DPAC leaders who proposed an excellent project so that I have learnt a lots and enjoyed working within the project. I am always thankful to administrative officers who have made all the complicated administrative procedures to become easy and smooth.

Last but yet importantly, I would like to express my utmost appreciation to my family in Vietnam, my husband, my lovely son Sam and my little newborn daughter Alisa. I am truly indebted to them for their unconditional supports, endless love, and encouragement. They are always beside me, and available whenever I feel down and need someone to share the burden of stress. They are the ones whom I want to care about most, the persons whom I want to bring all happiness to. I would like to dedicate this thesis to them whom I love most in my life.

Lan Anh Trinh
Västerås, December, 2021

Sammanfattning

Under årtionden har automatiska robotar som är förprogrammerade för att utföra repetitiva uppgifter i industriell produktion nått framkant inom teknik. Det kommer nästa utveckling med autonom kontroll, där en robot kan ha vissa nivåer av sitt eget beslut, det vill säga självstyrande, utan direkta kontroller från människor. Detta ger autonoma robotar i stor utsträckning tillämpliga inte bara inom industrin utan också i vanligt tillgängliga tjänster i vårt dagliga liv, till exempel självkörande bilar, automatiserad sjukvård eller underhållning. Ändåmåste en av ryggraden i robotsystemet, navigering och vägplanering, möta allt fler utmaningar, inklusive ostrukturerade miljöer, osäkerhet kring rörliga föremål, samexistera med människor och flera robotagenter. Syftet med ett pålitligt, dvs tillgängligt, tillförlitligt och säkert, vägplaneringssystem för att övervinna sådana utmaningar, föreslår denna avhandling att utveckla flervägsplanering tillsammans med algoritmer för att undvika hinder och överbelastning. Till en början föreslås ett nytt dipolflödesfält, som är konstruerat från ett flödesfält för att driva robotar till sina mål och ett dipolfält för att skjuta robotar långt bort från potentiella kollisionsriktningar. Algoritmen är effektiv vid implementering men kan övervinna nackdelen med konventionellt fältbaserat tillvägagångssätt, som lätt fångas av en lokal optimering av energifunktioner. För det andra utvecklas en mekanism för överbelastningskontroll med Petri net för att synkronisera rörelser hos robotar när de kommer in i ett kors eller smalt område. Olika Petrinät utvärderas för att hitta den optimala konfigurationen för att minska trafikstockningen genom möjliga konfliktregioner. I nästa bidrag

åtgärdas problemet med död- eller livlåsning av ett vägplaneringssystem. Lösningen baseras påmultipelvägsplanering där varje robot har alternativa vägar till målet. Alla robotar i samma arbetsutrymme kommunicerar med varandra för att uppdatera sina platser och vägar såatt lämplig konfiguration kan väljas för att undvika potentiella dödlägen. Algoritmen tar ocksåhänsyn till hinderundvikande såatt robotarna kan undvika inbördes kollisioner såväl som kollisioner med oväntade rörliga föremål som människor. Slutligen implementeras en decentraliserat algoritm för multipla banplanering för att hjälpa systemet att hantera vissa nivåer av fel, vilket händer när det centrala styrsystemet för robotar slutar fungera eller en del av kommunikationsnätverket mellan robotarna oväntat kopplas bort. De föreslagna tillvägagångssätten har utvärderats genom omfattande experiment för att visa deras effektivitet för att hantera kollisioner, trängsel och blockeringar. Implementeringen av algoritmerna har utförts påallmänt tillgänglig plattform, robotoperativsystem (ROS) och överförts till riktiga robotar.

Abstract

Over decades, automatic robots that are pre-programmed to perform repetitive tasks in industrial production has been reaching the cutting edge of technology. There is emerging the next development with autonomous control, where a robot is able to have some levels of its own decision, i.e. self-governing, without direct controls from humans. This brings autonomous robots extensively applicable not only in industry but also in commonly accessible services in our daily life such as self-driving cars, automated health care, or entertainment. Yet, one of the backbone of the robotic system, the navigation and path planning, has to face more and more challenges including unstructured environments, uncertainty of moving objects, coexist with humans, and multiple robotic agents. Aiming toward a dependable, i.e. available, reliable, and safe, path planning system to overcome such challenges, this thesis proposes the development of multiple path planning along with obstacle avoidance and congestion control algorithms. At first, a novel dipole flow field, which is constructed from a flow field to drive robots to their goals and a dipole field to push robots far away from potential collision directions, is proposed. The algorithm is efficient in implementation yet is able to overcome the drawback of conventional field-based approach, which is easily trapped by a local optimisation of energy functions. Secondly, a congestion control mechanism with Petri net is developed to synchronise the movement of robots when they enter in a cross or narrow area. Different Petri nets are evaluated to find the optimal configuration to reduce the traffic jam through possible conflict regions. In the

next contribution, the dead- or live-lock problem of a path planning system is addressed. The solution is based on multiple path planning where each robot has alternative paths to the goal. All robots in the same working space communicate with each other to update their locations and paths so that the appropriate configuration can be chosen to avoid potential deadlocks. The algorithm also takes into account the obstacle avoidance so that the robots are able to avoid mutual collisions as well as collisions with unexpected moving objects like humans. Finally, a decentralised multiple path planning algorithm is implemented to help the system to deal with some level of failures, which happens when the central controlling system of robots stops working or a part of communication network between the robots is unexpectedly disconnected. The proposed approaches have been evaluated by extensive experiments to show their effectiveness in addressing collisions, congestion, as well as deadlocks. The implementation of the algorithms has been performed on widely accessible platform, robot operating system (ROS) and transferred into real robots.

List of Publications

Papers Included in the Doctoral Thesis[1]

Paper A Toward Shared Working Space of Human and Robotics Agents Through Dipole Flow Field for Dependable Path Planning
Authors: Lan Anh Trinh, Mikael Ekström, Baran Cürüklü
Status: Published in Frontiers in Neurorobotics, volume 12, 2018.

Paper B Dependable Navigation for Multiple Autonomous Robots with Petri Nets Based Congestion Control and Dynamic Obstacle Avoidance
Authors: Lan Anh Trinh, Mikael Ekström, Baran Cürüklü
Status: Submitted to Journal of Intelligent and Robotic Systems, Springer, 2021. Minor revision.

Paper C Multi-Path Planning for Autonomous Navigation of Multiple Robots in a Shared Workspace with Humans
Authors: Lan Anh Trinh, Mikael Ekström, Baran Cürüklü
Status: Published in The 6th IEEE International Conference on Control, Automation, and Robotics, 2020.

Paper D Decentralised Multi-Robot Path Planning with Obstacle Avoidance and Congestion Control Constraints

[1] The included papers have been reformatted to comply with the thesis layout

Authors: Lan Anh Trinh, Mikael Ekström, Baran Cürüklü
Status: Submitted to IEEE Access.

Additional Peer-Reviewed Publications, not Included in the Doctoral Thesis

- ”Fault Tolerant Analysis for Dependable Autonomous Agents Using Colored Time Petri Nets”. Lan Anh Trinh, Baran Cürüklü, Mikael Ekström. The 9th International Conference on Agents and Artificial Intelligence, ICAART 2017.

- ”Dipole Flow Field for Dependable Path Planning of Multiple Agents”. Lan Anh Trinh, Mikael Ekström, Baran Cürüklü. Workshop on Shared Autonomoy, IEEE/RSJ International Conference on Intelligent Robots and Systems, 2017.

- ”Failure Analysis for Adaptive Autonomous Agents using Petri Nets”. Mirgita Frasheri, Lan Anh Trinh, Baran Cürüklü, Mikael Ekström. The 11th International Workshop on Multi-Agent Systems and Simulation (MAS&S’17).

- ”Dependability for Autonomous Control with a Probability Approach”. Lan Anh Trinh, Baran Cürüklü, Mikael Ekström. Newsletter in ERCIM News 109, Autonomous Vehicles.

- ”Petri Net Based Navigation Planning with Dipole Field and Dynamic Window Approach for Collision Avoidance”. Lan Anh Trinh, Mikael Ekström, Baran Cürüklü. Published at 6th International Conference on Control, Decision and Information Technologies, 2019.

Contents

I

Thesis

Chapter 1

Introduction

There has been an undeniable inclination of the revolution in robotics and autonomous control over the last decade. The trend has started since the new definition of Industry 4.0 [1] based on the foundations of robotics, Internet of thing (IoT), wireless communication, and artificial intelligence was introduced. Furthermore, the pandemic with Covid-19 increases the needs of autonomous robots and virtual communications to reduce direct contacts among humans. So far, numerous intelligent robotic systems have been developed to support humans in a variety of tasks in both daily life such as healthcare services, smart home, autonomous driving, as well as industrial environments such as warehouse systems, cargo robots, or robotics hands. Generally, an autonomous system primarily composes of four specific yet interconnected components in which they are integrated and influenced by the design of the system architecture varying for different applications, i.e., (i) sensors and sensor fusion, (ii) modelling and control, (iii) map building and path planning, and (iv) decision making and autonomy.

Obviously, artificial intelligence is now a crucial part of an autonomous robot system to help it to deal with decision making without receiving direct commands from humans in some extends. To fulfil the mission, particularly

for autonomous mobile robots, path planning and navigation is one of the core components, which needs to be efficiently designed for the controlling system of robots. There are certain issues (Figure 1.1) related to solving the path planning problem, including how to generate a natural move, i.e the moving trajectory of the robot is smooth and has few turns; multiple robotic agents, i.e how the robot can coordinate with others to avoid mutual collisions; environment uncertainty where the robot needs to handle unpredictable moving objects such as humans; deadlocks in which the robot is able to avoid/escape a loop trajectory to approach its defined goal; lastly, efficiency means that the robot is able to choose the most optimal path to accomplish its navigation tasks with minimal energy consumption.

Whatever solution is used to address any of those issues, the main aim has led to the development of a dependable, i.e, availability, reliable, and safe, algorithm for a navigation system [2, 3]. Hitherto, dependability has been well studied in the literature to introduce a pathway to understand faults and to proactively prevent them to happen inside and outside a system. Targeting toward a dependable path planning algorithm makes it more widely acceptable for both daily life and industry applications. To reach there, first the analysis of what are the root that could cause the failures of a path planning algorithm is performed. The assumption is that the robot is equipped with sufficient hardware and software to know the relative location of the robot with respect to others and to destination. The robot must have the map of environment and sensors to be able to detect surrounding obstacles. Finally, it is able to communicate wirelessly to share information among each others in the same working space. Those requirements are kinds of standard in robotics where all functions now are easily reachable by just using embedded devices and computational board. Back to the main focuses of this study, the first question is how to ensure efficiency of the path planning algorithm. The answer to this question relates to the amount of energy spent by a robot for a moving task. It is obvious that routing a robot through a zigzag path is not optimal as the activity of slowing down the robot and speeding it up again usually consumes energy unnecessar-

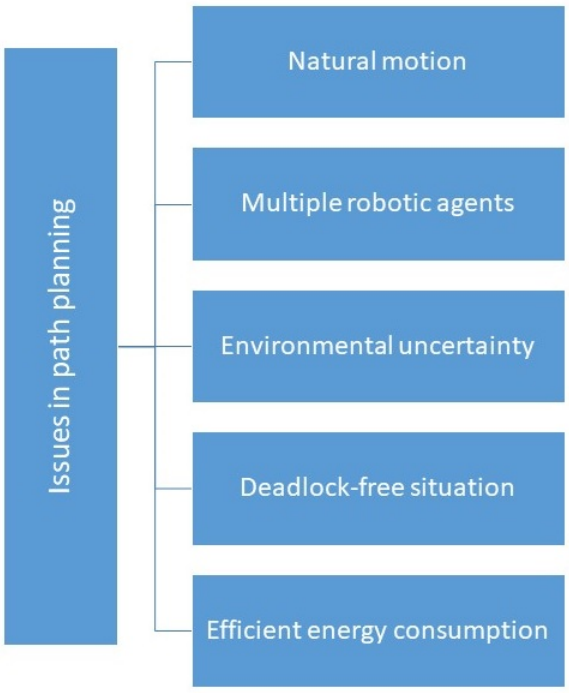


Figure 1.1: Certain issues in navigation and path planning.

ily. Thus, smoothening the path plays an important role here to address such a problem. Second, the path planning algorithm needs to be reliable to correctly drive each robot to its goal. The continuity of the service is also important to make it always available to the robot system. However, a robot while trying to avoid moving obstacles and other robots could face live- or dead-locks which make it no longer be able to move. The problem happens if the robot thinks a single global path to the goal and only takes into account the local information in a short range to decide the next move. Planning multiple paths and utilising global information shared by all robots in the working space are the key to help

robot to predict and avoid the congestion region. Finally, it is critical to have a safe path planning algorithm to ensure no mutual collisions among robots and collisions between robots and humans. This is to avoid any fatal consequence on humans, a robot itself, and surrounding environments.

With regards to above analyses, the main contribution of this thesis is to develop a dependable path planning algorithm for a system of multiple robots by focusing on collision avoidance, congestion control, and centralised as well as decentralised multiple path planning. The work has started with the introduction of an effective path planning frame work based on dipole flow field, which includes the static flow field to drive the robot to its goal and the dipole field for obstacle avoidance. The proposed algorithm has low computational complexity yet is effective to mitigate the local optimisation of conventional field-based approach. The path driven by the flow field is suitable to plan dynamic motions of mobile robots and has been proved with real implementation. Continuously, the obstacle avoidance is advanced into the next level with congestion control. The basic idea is that routing too many robots into a same place could lead to a congestion that robots must go around and around to avoid collisions. Sometimes, they block each other inside a narrow area. A Petri net, an efficient tool to solve resource conflict for a dependable system, has been utilised in this work to synchronise the movements of multiple robots through a region or a cross. An optimisation problem is formulated to model the traffic of multiple robots where its solution returns the most suitable Petri network to control robots' movements. Finally, in order to tolerate up to some level of failures of a navigation system, a centralised as well as decentralised framework with multiple path planning is proposed. Each robot will has several possible paths to the goal which can be the shortest path found by conventional searching algorithms or any longer path but to provide alternative selection for a robot. The robots communicate with each other to negotiate the moving paths so that they are able to proactively avoid the conflicts that may cause a potential deadlocks in future. The obstacle avoidance is also taken into account to help robots to avoid collisions with any moving obstacles on the moving way. In overall, the

summary of the dependable multiple path planning system proposed in this thesis is depicted in Figure 1.2.

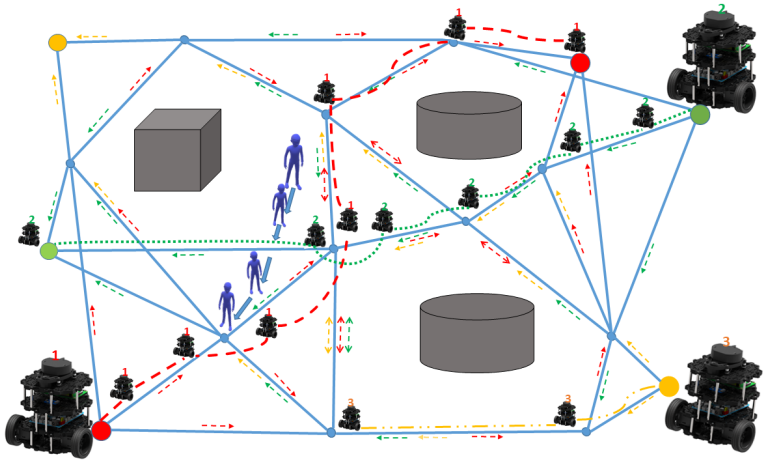


Figure 1.2: Overview of the proposed path planning algorithms. The multiple potential paths of three robots (marked with red, green, and yellow colours) to move from one vertex of the graph to other are described by dash arrows. The actual moving trajectories are chosen to minimise the conflicts and collisions and expressed by bold lines without arrows.

Thesis Overview

This thesis is divided into two main parts.

- The first part of the doctoral thesis is organised as follows. Chapter 2 presents the basic foundation of the research work, including the definition of dependability, and the key features of dependable path planning. Chapter 3 summarises the state-of-the art techniques for dependable path planning for autonomous robots, congestion control and multiple path planning. Chapter 4 presents the pathway to conduct this research work

from initialise the target of the research to define the research questions, and hypotheses. Continuously, Chapter 5 summarises the major contributions of this work and Chapter 6 concludes the thesis with some discussions on the development of path planning algorithms at present and in future.

- The second part of this thesis is a collection of four articles (paper A-D) presenting the main contributions of the thesis in details.

Chapter 2

Dependable Path Planning

This chapter provides the backgrounds of dependability and the proposal of a dependable path planning system. The chapter is divided into three main sections. As first, the general definition of dependability and its role in the autonomous system are introduced in Section 2.1. Correspondingly, the overview of how those dependable properties are applied in the path planning algorithm is described on Section 2.2. Finally, the implementation for dependable path planning with the uses of multiple paths, obstacle avoidance and congestion control is introduced in Section 2.3.

2.1 Dependability and Autonomous Robots

Recently, autonomous robots are being widely used in numerous areas in which the robot either collaborates with other robots and humans to complete a task or even performs the task alone. For those robots to be able to work with humans, safety is the most important factors to avoid possessing any dangers. For those robots operating by themselves, they must be able to handle failures without a humans assistant. The above issues raise new research challenges in robotics, the dependability of robotic systems. Although the implementation of a de-

pendable system increases the cost in the short-term, it gains the sustainability of the system over long-term runs.

Originally, dependability is defined by Avižienis et al. [2] as the ability to provide services that can be trusted. Basically, to assess the dependability of the system, five main attributes *availability, reliability, safety, integrity*, and *maintainability* are introduced including as given in Table 2.1.

Table 2.1: Dependable properties.

Availability	Be available or ready to provide services at any time
Reliability	Continuously provide of correct services within a period of time
Safety	Avoid any fatal consequence on operators, other robots, and environments
Integrity	Avoid improper system changes, like unauthorised access, or modification of program
Maintainability	Support the processes of modifications and repairs

In overall, the autonomous robotic system must be able to recognise, work with, and adapt to humans and other robots' behaviours in a possibly unstructured and unknown environment. These possess challenges corresponding to different aspects to implement the dependability of an autonomous system such as:

- *Learning and adaption*: Adaption of robots to unknown conditions in new environments with limited learnt knowledge.
- *Modelling, analysis, and simulation*: Development of a model simulation to analyse and verify the dependability of robot systems.
- *Control and planning*: Robot control under some levels of uncertainty and co-operation with other robots.
- *Perception*: Perception of environments in hazardous and complex situation.

- *Human-robot interaction*: Collaboration and interaction between humans and robots.

This thesis mainly addresses the challenges related to the **navigation control and planning** of autonomous systems and focuses on the three main properties including **availability**, **reliability**, and **safety**.

In order to implement the dependable properties [2, 3], threats are defined to understand any factor which affects system dependability. In overall, threats include failures, errors, and faults. The failure happens when the provided service is deviated from the correct service due to an error. As the cause of the error is fault, fault is considered as the root of every failure happening both inside and outside of the system. Therefore, in order to enhance system dependability, the system must be prevented from faults with the introduction of means. There are four certain means including fault prevention, fault removal, fault forecasting, and fault tolerance. Fault prevention requires the use of appropriate development methodologies, standard implementation techniques, and good practices to design the system; Fault removal utilises verification tools through a log-file to identify, diagnose, and remove faults; Fault forecasting estimates the possible incidence and what can be the consequence of faults with safety or risk analysis; Finally, fault tolerance is implemented with redundant hardware or software to provide recovery mechanisms once a system fails to complete its task. The last mean is usually applied during the operations of a system.

The taxonomy showing the relationship between dependability and its components is given in Figure 2.1.

2.2 Dependable Path Planning

Path planning, basically, is defined as finding a sequence of points to construct a free collision way for a robot to reach a desired goal. In general, the path planning system is divided into global path planning and local path planning. The global path planning system of the robot perceives information on the

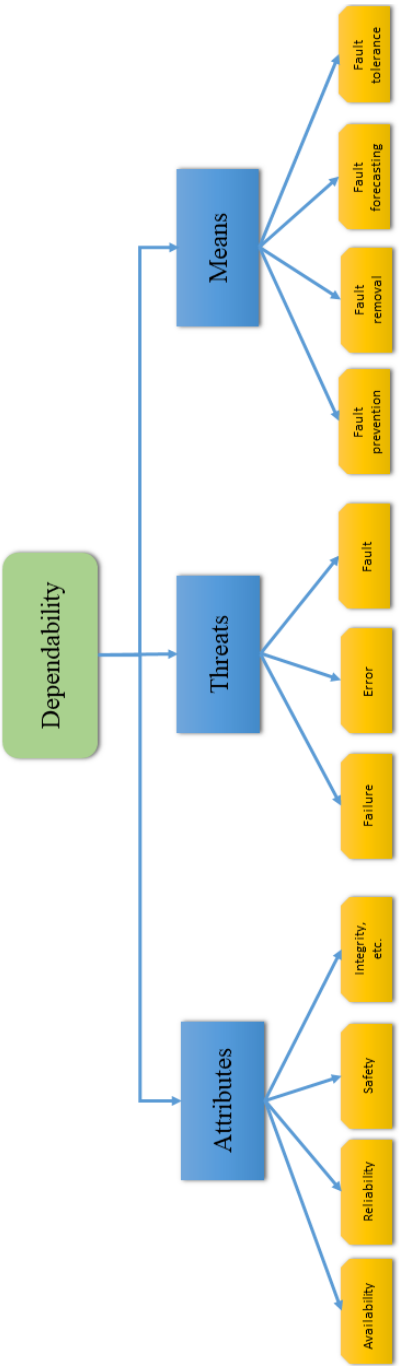


Figure 2.1: Dependability taxonomy.

surrounding environment to generate a route from a starting point to a predefined goal. Meanwhile, the local path planning controls movement of the robot along the global path to avoid collision with all the obstacles within a local area. Conventionally, the global path planning mainly applies for static environment, i.e., the presence of other moving objects such as other robots and human is not taken into account when the global path is established. There are numerous algorithms used for global path planning, however the graph/map/-grid search-based algorithms are the most popular techniques for searching in a map. For the local path planning where the robot has to deal with both following the global path and collision avoidance.

In this thesis, a dependable path planning algorithm is proposed with the aim to cope with both global and local range and to avoid both static and uncertain dynamic obstacles like humans. As aforementioned, three main dependable attributes are focused including availability, reliability, and safety. Given the robot an interval time to finish a navigation mission, those three attributes are expressed as following:

- Availability: The availability is measured by the continuity of the path planning algorithm. Therefore, the availability of the navigation system is measured by the probability that the system is functioning correctly at an instance time.

- Reliability: This attribute presents the continuity of correct services. This means the path planning algorithm is expected to operate without interruption. Thus, the reliability in the given interval time of the navigation system is measured by the probability that the robot is still running under the control navigation system without any failures.

- Safety: This attribute is assessed by the ability that the navigation system can avoid any catastrophic consequences on the users, other robots and the environments. This means during the given interval time for the robot to complete the navigation task, the navigation system enables the robot to avoid all obstacles including moving objects such as humans to reduce

the danger to the environment, also the robot is less interrupted while operating.

To aim toward a dependable path planning algorithm, at first, it needs to be ensured to be implemented on a good platform with ready functions of coding standardisation, unit-testing, building system, documentation, etc. By choosing the well developed open-source framework ROS [4], and adapting its path for implementation, the necessary tools for the first two means, fault prevention and fault removals, are basically realised in thesis. This thesis mainly concentrates on the development at algorithm levels to address the fault prediction and fault tolerance. They are expanded into specific aims for the path planning algorithm. First, the algorithm is able to drive all robotic agents to reach their goals with optimal paths correctly. Here, the optimal path is defined as a path with a minimum cost calculated based on one or some the elements including the total length of the path, the energy used by a robot to complete the path, the complexity of the terrain over the path, etc. Second, for safety reasons the robot needs to avoid collisions with other robots and humans in its working space. Lastly, the robots are also required to complete the moving tasks. This means that if the robots are not able to reach the goals or take a long time to do so, it is considered that the robots are in the deadlock situation and failed to complete a path planning task. It is obviously that should potential failures of a path planning algorithm be proactively located, they can be mitigated. The congestion control is developed in this thesis as a new mechanism to reduce traffic in congestion areas, so that several robots are not routed into a same narrow place, which possibly leads to a deadlock. Meanwhile, during operations of robots, fault tolerance enhances the system ability to handle with failures by using backup solution. Adapting fault tolerance into a path planning algorithm, this thesis proposes the uses of multiple path planning to allow each robot to have several options to reach to its goal and the communication of all robots in their work space to find optimal path assignments. The overall dependable path planning approaches presented in this thesis include obstacle avoidance, congestion control, and multiple paths.

2.3 Dependability with Obstacle Avoidance, Congestion Control, and Multiple Path

In summary, the conventional global path planning provides the basic dependable properties of a navigation system with the function to drive a robot to a goal within the map of static objects. Obstacle avoidance enhances all safety, availability and reliability by avoiding the collisions with moving obstacles that can lead to the stop of navigation services. The congestion control and multiple path drive the robots to less crowded and less risky areas, thus indirectly help to improve all three dependable properties. Finally the extra implementation of the system with decentralised manner reduces the dependency of robots to one central node to enhance the availability and reliability in general. The dependable attributes are mapped into different component of path planning algorithms in Table 2.2.

Table 2.2: Dependability attribute map.

Attribute	Obstacle avoidance	Congestion control	Multiple path	Decentralised manner
Availability	✓	✓	✓	✓
Reliability	✓	✓	✓	✓
Safety	✓	✓	✓	

Chapter 3

Related Works

As aforementioned, the requirements that an autonomous navigation system needs to be dependable, i.e. safe, available, and reliable, are becoming more and more important. This chapter provides overall literature of related researches. The dependability for autonomous robots is surveyed in Section 3.1 while the introduction of path planning algorithms is given in 3.2. Continuously, other works about obstacle avoidance are surveyed in Section 3.3, and about congestion control and multiple path planning are given in Section 3.4 respectively.

3.1 Dependability and Autonomous Robots

Over decades, separate dependable attributes of an autonomous system have been developed. For example, an intelligent home care robot, Care-O-bot, was introduced by Birgit et.al [5] for elderly people assistance. The robot was mounted with extensive sensors to detect motion, and also to prevent accidents caused by a person hitting the robot with different levels of safety. Meanwhile, a dependable platform has been concerned in industrial robots to create collaborating working environment between human and robot in manufacturing

factories. For instance, ABB has been working on an open robotics platform [6] with analysis tools in order to monitor and find the failure robot causes with a testing procedure. Close to the works presented in this thesis, there has been a rise on the development of self-driving cars from big companies like Tesla, Google, Uber, or Volvo. To minimise the risks of causing accidents to human, autonomous cars would have to tested to be driven in different environments like city traffic, busy highways, etc. to demonstrate a safe and reliable self-driving system [7, 8].

Regarding definition from software perspective, the dependability of a system is evaluated by attributes including availability, reliability, safety, integrity, and maintainability [9]. The dependability of a system can be assessed by one, several, or all attributes mentioned above. The researches on dependability have shown that faults are the roots of failures occurring inside or outside of autonomous system. Thus, different means are developed to prevent faults and enhance the tolerance of a system to failures. Fault prevention focuses the use of appropriate tools. This can be accomplished by good implementation techniques, good tools, and the component-based designed of a system such as LAAS [10], RAX [11] or standardised middleware like robot operating system (ROS) [4], OROCOS [12], etc. Fault removal deals with identifying, diagnosing, and removing faults. Fault identifying is performed with verification tools including dynamic verification (run test to find faults through log-file and analysis) [13] and static verification [14] (system analysis, model checking, etc.). Besides, during the operation stage, all failures must be logged for the maintenance cycle. Fault forecasting deals with the prediction of the potential future incidence and its consequence. If the robot system is described by unified modelling language (UML) [15], then the safety analysis is studied with Hazard and Operability Studies (HAZOP) [16] or Fault Tree Analysis (FTA) [17]. Other common approaches are Bayesian network and Petri net (PN). Finally, fault tolerance is implemented by using redundant hardware and software [18, 19, 20].

3.2 Global Path Planning

First of all, conventional algorithms about path planning are introduced in this section. Many of them have focused on searching to find a path from a source to a destination in static map without considering moving obstacles. One of the most conventional yet still effective approaches for the navigation of an agent in a large map is related to Dijkstra-based algorithm [21], its extension of A* searching algorithm [22, 23], and incremental search [24]. In details, the A* algorithm improves the Dijkstra's algorithm by approximating the cost-to-go function with heuristics to reduce the expansive of the searching tree to the goal. Meanwhile, incremental search algorithms seek for the shortest paths by utilising the results of similar searches to make the search faster, instead of solving each search problem separately. By applying incremental search on top of the A*, Koenig et al. [25] developed lifelong planning A* (LPA*) as an initial variant of A*, in order to address path planning for dynamic graphs where edge costs are updated. In the D* algorithm [26], incremental search is applied to repeatedly update the shortest paths between the current position of a robot and a goal, during the travelling of robots to approach to the goal. Koenig and Likhachev [27] improved the D* by LPA* to have D*-Lite and alternatively Sun et al. [28] developed dynamic fringe saving A* to reuse the OPEN and CLOSED lists from previous A* searches.

Although different variants of A* are able to address a graph change due to the moving of a robot to a new vertex, or the incremental updates of edge costs, there are lacking particular versions of A* to deal with moving obstacles. One of the approaches have been developed by Hu and Brady [29] was to model the uncertainties of mobile obstacles using probability map. However, the complexity of path planning significantly increases when the cost of the edges on the graph are presented by random variables. Therefore, the most common solution is to keep a robot moving on global path but the obstacle avoidance is handled separately.

3.3 Obstacle Avoidance

The limitation of global path planning to deal with dynamic moving obstacles is addressed by obstacle avoidance, which is considered as path planning within a local range. Conventionally, a field-based approach is of the way to combine the global path planing with obstacle avoidance. The solution is usually constructed by the combinations of two fields including a repulsive field to push a robot away from the obstacles, and an attractive one to pull the robot towards the goal. The Voronoi diagram was proposed by Ok et al. [30] to partition the working space into regions which are close to static obstacles. Correspondingly, the repulsive field was built along with the attractive field to route a robot to it goal. Meanwhile, the physics model of heat flow was used by Wang and Chirikjian [31] and Golan et al. [32] where obstacles were presented by hot obstacles and the goal was the cold one. Solving partial differential equations of heat exchanges among objects has revealed a way to drive a robot to a goal without colliding with obstacles. Obviously, the biggest limitation of using the field-based approach is the trap into local optimum due to the cancellation of repulsive and attractive field. In consequence, a robot stops moving at local optimal point where no forces are generated. Attempts have been introduced to reduce the local convergence with extra elements. For instance, García-Delgado et al. [33] customised the repulsive field with regards to the angle between an attractive force to an obstacle. The basic idea is to mitigate the cancellation of the two opposite attractive and repulsive forces once present. Unfortunately, velocity is one of the key factor to avoid collision with moving obstacles but it has not been utilised effectively in most of field-based approaches.

So far, there has been another direction to use velocity constraints to seek for a collision free path. Particularly, by knowing the current location of robots and their moving directions with velocities, the collision regions are defined in velocity space (VO). Owen and Montano [34, 35] extended this definition of VOs into the obstacle avoidance of robots. Similarly, Damas and Santos-

Victor [36] developed the forbidden velocity zones with the bounds on robots' velocity so that a robot adjusts its moving speed accordingly to avoid obstacles. In addition, other extensions of VOs have been investigated such as the integration with acceleration in the works presented by Berg et al. [37, 38] and by Wilkie et al. [39], or the consideration of different robot shapes tackled by Lee et al. [40]. However, the complexity of the VO-based approach increases significantly once static obstacles are included. In particular, each static obstacle is presented by a VO where its shape needs to be taken into account with representation as a polygon or a similar kind of complex geometric object. Using velocity control sometimes results in oscillatory motion once robots come back to the preferred path after escaping from a collision area. Finally, a fast collision avoidance is necessary under urgent situations, for example, when a human approaches closely to the robot.

In order to address the above mentioned issues, in this thesis, a new obstacle avoidance with dipole flow field has been proposed. Obviously, the lacking of a mechanism to drive a robot to follow the global path to the destination could lead to a trap by a local optimum. The dipole flow field integrates the global path planning into the navigation field to address such an issue. For global planning, the method applies any-angle path planning algorithm of Theta* [41] to generate smooth paths with few turns, from a starting point to a goal for a pool of agents. Although different A* variants of any-angle path planning have been proposed, such as A* post smoothing, block A* [42] and field D* [43], the Theta* is able to provide the optimal path with effective implementation once compared with others [44]. As the computations of the Theta* algorithm is costly, the a static flow field along the planned path is defined to pull the robot back to the continue reaching to the goal if the robot has a small deviation from the planned path. Meanwhile, the dipole field is developed to help robots avoid collision with others and humans the shared working space. So far, conventional field-based methods generate pushing forces with respect to the location of robots. In this thesis, the moving directions and velocity amplitudes of robots are taken into account in dipole field to push robots far away

from prospective collision area. Last yet importantly, the proposed dipole flow field is simple but effective to provide a quick response of a robot to sudden events like the appearance of an unexpected moving obstacles in a very close distance.

In overall, different obstacle avoidance mechanisms help to gain the dependability of a navigation system to mitigate the effects from external factors like moving obstacles on its performance. Continuously, the availability and reliability of a navigation system is improved further with proactive manner using congestion control as presented in the following section.

3.4 Congestion Control and Multiple Path Planning

As mentioned in the previous chapter, fault tolerance is one of key mean to implement dependable system with redundancy. However, the redundancy is also needed to be combine with an analysis method to proactively predict what could wrongly happens to the system. Therefore, congestion control is combined in this work to help the system to analyse surrounding environments and make a right decision.

Commonly, the term of congestion control has been widely applied in transportation to avoid traffic jams [45]. To take advantage of this into robotics, it has been used to monitor and control the moves of the swarm robots [46]. In order to extend the definition of congestion controls in path planning, in this work, PN has been utilised to organise the path planning tasks of multiple robotic agents solve conflicts and collisions among them. The PN framework is chosen because it provides solution for fault analysis and fault prevention in both development and operational stages of a system. Related works of PN with dependability are found in [47, 48, 49], PN for system modelling and analysis in [50, 51, 52, 53, 54], PN with collaborative robots in [52, 55, 56, 57], PN to design and implement robots in [58, 57, 59, 53], PN and fault tolerance in

[49, 60, 15], and PN with ROS in [61].

Aforementioned, the main aim of this thesis is to develop a dependable navigation system for multiple autonomous robots. To avoid congestion leading to deadlock situation where the robot is blocked by human or other robots to reach the goal. Besides, to make the navigation system more dependable, i.e., fault-tolerant system, a proactive multiple path planning algorithm is proposed in which the autonomous robots are equipped with multiple global paths to the goal. Multiple robots are able to communicate to exchange information about the planned paths and negotiate to find the optimal paths for all robots in a global scale.

For the graph-based solutions, the robots move on a connected graph from a vertex to its neighbours in one time step to reach their goals. A conflict happens when two robots occupying on a single vertex at the same time and the main aim of the path planning is to find a conflict-free solution satisfied by all robots. An extra cost function, namely sum-of-cost, like the total maximum time for all robots to reach their goals (or the sum of all path cost) is introduced as an optimal condition for the search. As the problem is shown to be non-deterministic polynomial-time (NP) hard [62], numerous approaches have chosen to seek for a close optimal solution to reduce processing time. A*-based optimal search finds a non-conflict solution among all combinations of assigning k-agent into the graph. To avoid the exponential grows of the state-space with respect to the number of agents, different methods have been applied, e.g. independence detection (ID) [63] to include robots only when necessary or operator decomposition (OD) [63] to treat a move as an operator to control a single robot at a time. Alternative to A*, the increasing cost tree search (ICTS) [64] proposed two-layers including high-level and low-level searching where the lower is used as a goal test of the higher. Another solution different from A*, conflict based search (CBS) is introduced by Sharon et al. [65]. In CBS, the agents are constrained by a triple of parameters including agent, occupying vertex, and time step. It means that the agent at the particular time step is refused to occupy an occupied vertex. The path is found only if all agent's

constraints are satisfied. The searching is completed when the paths for every agents are resolved.

Beside above solutions, there have been suboptimal solutions for multiple agent path planning. For instance, hierarchical cooperative A* (HCA*) [66] introduced a reservation table which is used to store the path assigned into an agent. The other agents according to their priority in the group search for their paths not registered in the reservation table and update the table accordingly after the paths are found. In an improved version of HCA* like Windowed-HCA* (WHCA*) [66], the reservation table is only applied for a limited time slot, i.e., window, when the other agents are rejected to reserve to the table. Later, in the work of Bnaya and Felner [67], a dynamic window focused around conflicts and agents likely to be involved to a conflict are prioritised to be processed next. In overall, the heuristic search A* and its variants are still costly computational solutions.

There have been researches developed to reduce the running time of the search-based algorithms with rule-based algorithms. Specific rules are defined for the movement of the agents to reduce searching time. Yet, the resulted paths from the rule-based algorithms are not always optimal. Alternatively, in the work of Yu and Lavalle [68], the path planning problem for multiple agents is modeled as a network flow and the collision-free paths are found by the integer linear programming (ILP) solver.

Most of conventional works on multiple path planning are based on an assumption of a working environment without the presence of human. Due to the safety reasons, the operation of robot will be terminated when a human enters the working zones or crosses the moving trajectories of robot. In this thesis work, a new method of multiple path planning is proposed to consider the human and other uncontrolled moving objects as factors into the path planning problem. This help to enhance autonomous functions of robot navigation to allow more flexibility of robots to continue working even with the presence of other robots in unfamiliar environment.

Chapter 4

Research Overview

The research presented in this thesis has been divided into four main stages (Figure 4.1) which reflect the step-by-step evolution to build up the whole system.

In the beginning of research, the investigation on a dependable path planning algorithm was conducted. The details of dependability properties and their roles in autonomous control system have been reviewed in comprehensive surveys, which result in the research of using Petri net to analyse the failures happened inside a system of multiple robotic agents. The study was presented in a conference paper (ICCART, 2017). Based on that, the dependable properties suitable for a navigation system are selected where a dependable, i.e., safe, reliable and available, path planning algorithm is proposed based on a novel dipole flow field which consists of a static flow field and a dynamic dipole field. The static flow field is to drive the robotic agents to the goals with less requirements to update the global path, meanwhile the dipole field plays an important role to prevent collision with other agents and moving obstacles appearing in the working space. The idea was initially presented in a workshop paper, later extended into a full paper [Paper A]. After the principle of the algorithm is evaluated through extensive experiments, its implementation on the

middle-ware ROS framework has been performed to transfer from theory into a practical system. Eventually, the implementation was done on Husqvanar Research Platform within the scope of UNICORN master project.

Relying on a single path could lead to deadlock or congestion when multiple robotic agents are routed into a narrow area. Therefore, in the second stage, a proactive multiple path planning algorithm has been proposed to avoid deadlock situations with congestion control. In this work, the global paths of the robots are formed as a graph of a Petri net network. The movement of the robot along its global path is synchronised into the movement of a token in the network. Moreover, a communication channel is also established to help the agents to share their information of positions and velocities to each other. Hence, the control place, i.e., the place where the potential congestion happens, in the Petri net of the agents is estimated, so that the agent can predict when the congestion happens before hand. Thank to the integration with the dipole field, the agents are able to avoid collision at the control place of the Petri net. The result of the work initially is presented in a conference paper published at CODIT, 2019. Later, the proposed algorithm is extended into a full paper [Paper B]. In this paper, each robot is equipped with different paths from starting point to its goal. The travelling time of all trajectories of the robots are analysed by analysing the Petri net. The most optimal configuration of paths for the robots is chosen which is based on the criteria for congestion avoidance.

Finally, an optimisation problem for multiple path planning has been proposed to optimise the paths for all the robots with obstacle avoidance constraints. The evaluation of the proposed method is presented in [Paper C]. As the above proposed multiple path planning has been evaluated in a centralised manner, the interrupt of the centre node could cause the system failures. A decentralised approach is proposed to cope with such a problem, making the robotic agent less dependent on each other, aiming at a self-governing system. The result has been condensed in a journal paper [Paper D] with the implementation on Turble bot 3 to evaluate the presented method.

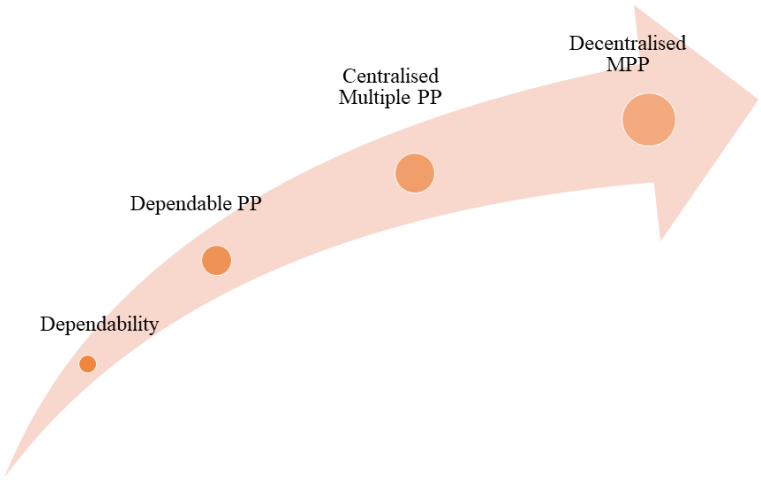


Figure 4.1: Comprehensive research flow.

4.1 Research Methodology

The path way to conduct research presented in this thesis follows inductive reasoning methodology. First of all, the overall objectives of the research are defined. With regards to the goals, the comprehensive literature review of previous works related to the research topics are studied. The research questions are defined based on the analyses of pros and cons of different approaches and of the missing pieces to be solved to achieve the main goals. Continuously, the hypotheses are stated to define what are expected to be evaluated with the ideas/algorithms proposed by the researches. In the next steps, the new ideas are invented and deployed to answer research questions. The experiments are performed to test the developed algorithms with regards to the defined hypotheses. Once the cycle of researches is finished, the limitations of the works are evaluated again to formulate the next iteration of researches. The research progress is therefore performed a step-by-step to build up the complete solution.

Along the pathway to conduct my research with inductive methodology, I always present the ideas of new theory in conferences to get comments from research communities to improve the contents of the work. Once the preliminary results are evaluated, the complete studies with more expensive experiments are performed to report the results into a journal before the next iteration of my research is circulated. In the implementation pathway, after a successful test in a comprehensive simulation, i.e. Gazebo, the real implementation is transformed into a real robot. The robotic platform, i.e. ROS, is chosen in both simulated and real environment to reduce the implementation efforts. My research strategy is summarised in Figure 4.2.

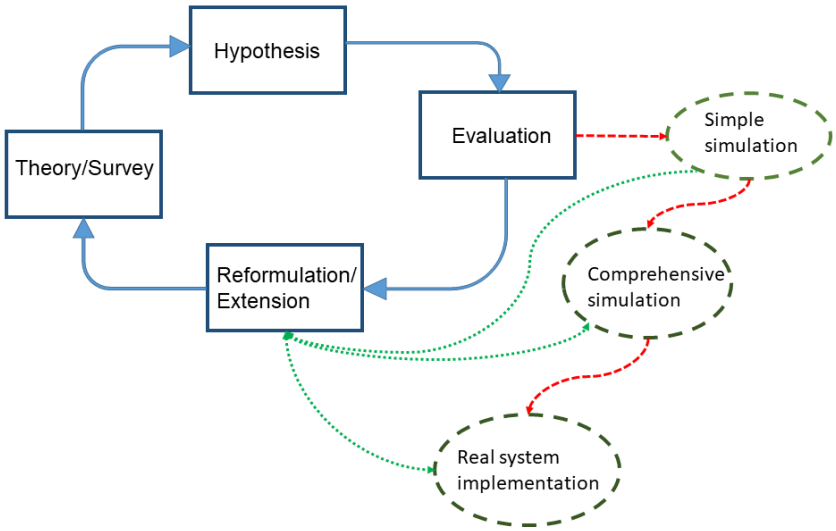


Figure 4.2: Research methodology.

4.2 Research Goals

In overall, the goal of the research works presented in this thesis is to develop a dependable path planning system for multiple robots to navigate them to reach the goals, assuming that the robots coexist with users/operators on the same working space. With regards to availability, reliability, and safety as the main dependable properties, the aims consist of driving all robots to the goals correctly, minimising the possibility and factors that could cause the stopping of the services like the live- and dead-locks, and finally, avoiding collisions among robots and between robots and humans. The hypothesis are defined at each step of my research accordingly.

4.2.1 Research questions

In the initial phases of this research, the research questions mainly focus on how to define the dependability of an autonomous system and then how to develop a dependable path planning for multiple robots system. Continuously, the ideas of obstacle avoidance, congestion control, and multiple path planning have been formulated. At every stages, the research questions (RQs) are raised accordingly:

1. RQ1: *How to define a dependable path planning framework for multiple robotic agents working in a sharing environment with human subjects ?*

   The answers to this question help to define the basics of a dependable system to extend them into path planning algorithms. This includes the theory of dependability, the means to implement a dependable system, the essential components of a path planning system, and the effects of multiple humans and robots on the sharing space into the design of path planning system.

2. RQ2: *How to develop a path planning algorithm to provide safe and reliable moving paths for multiple robots?*

RQ2 is a follow up of RQ1 to deploy dependability properties on a path planning algorithm. The questions are put in a context where humans and robots are living together and safety is a critical factor needs to be ensure to no serious collisions between them. It is noted that the positions of human subjects are estimated from the observation of sensors, thus some levels of uncertainty should exist. The complexity of environment is also taken into account to reach a robust solution that is not limited to a restricted area. Effectiveness on energy consumption is an extra factor to be considered in a path planning algorithm to target sustainable and environmental friendly solutions for long-term uses.

3. RQ3: *How to avoid a deadlock situation through multiple path planning and congestion control?*

   The aim of this question to enhance a path planning algorithm with fault tolerance. This means if a robotic agent fails to do a moving task to the target (deadlock situation), there should be a backup plan for it to recover its navigation activities. The question are to how to formulate the multiple path planning and how to use congestion control to help a group of robotic agents to proactively synchronise their movements through a cross intersection to avoid congestion.

4. RQ4: *How to integrate the proposed path planning algorithm on real robots? What case studies with real autonomous robots can be used to test the whole system?*

   Many challenges related to a realistic environment need to be concerned. They include how to choose a right framework to utilise existing developed solutions, how to detect and monitor the moving trajectories of humans and other objects, how to build a map and localise the robot in there, how to develop a communication method to share information among robots etc.

4.2.2 Hypothesis

At the beginning of this research, the basic framework for a planning algorithm is investigated along with the new navigation method for multiple robots. Aiming at the simple yet effective method to drive the robots to goals safely, the dipole flow field is invented. An any-angle-path planning with Theta* algorithm is employed to initialise the paths from a starting point to a goal for a set of robots. To deal with the movements of the robots, a static flow field along the configured path is defined. This field is used by the robots to navigate and reach their goals even if the planned trajectories are changed. In parallel, a dipole field is calculated to avoid the collision of robots with others and human subjects. In this approach, each robot is assumed to be a source of a magnetic dipole field in which the magnetic moment is aligned with the moving direction of the robot. The magnetic dipole-dipole interactions between these robots generate repulsive forces to help them to avoid collision. Note that the collision happens if the distance between a robot and the target is smaller than predefined thresholds. The first hypothesis is stated to evaluate the proposed path planning approach as follows:

Hypothesis 1: *Using the path planning algorithm as the combination of the dipole field and flow field, it is possible to correctly drive a robot to its desired goals and avoid the collisions of the robot with humans, other robots and obstacles on the moving way.*

Conventionally, path planning algorithms have addressed the navigation problem for multiple robots with a single path planning. Relying on a single global path planning could lead to a deadlock situation where a robot takes a very long time to reach a goal or even is not able to do so. This happens in the case of multiple robots moving in a narrow area. Since the local path planning to avoid obstacle only takes into account the collision of other robots in the vicinity, a robot must turn back to the predefined path to reach a goal. However, if two robots are routed through a very narrow area, like a corridor, and enter it through two opposite sites, robots may face a deadlock situation where they repeat the same moving trajectories through that area again and

again without finding the path to the goal. Considering human as a big obstacle due to safety reason could block a robot to enter a small way to follow its global path. Therefore, it is important for a robot to prepare different solutions to reach a goal and the robot needs to proactively switch between solutions whenever necessary to avoid dead lock situation. With regards to the studies about PN and the new path planning algorithm with dipole field, the research in this proposal is continued with the uses of PN for congestion control. Obviously, the navigation of several robots into one narrow area may make them to block each other and harder to find the way out from the area or to take longer time to reach their goals. To address the congestion of routing multiple robots into one place, a group of robots should find effective routes to avoid conflicts with each other. As PN is an effective tool in dependable autonomous control to resolve conflicting problems, it is able to apply PN to plan the paths of multiple robots with a delay to avoid routing many robots into a same place so that the robot may not need to turn around with longer paths to approach to their goals.

The next hypothesis to be tested is formulated as:

Hypothesis 2: *Using multiple path planning with PN to synchronise the movements of robots to cross through an intersection or narow region, it is able to avoid the deadlock situation or the livelock situation, i.e. agents "dance" back and forth when directly opposing each other.*

The multiple path planning using PN to resolve traffic jams of multiple robots is centralised, meaning that if the central controlling node is offline, the whole system stops working. In order to enhance the fault tolerance, it is important to decouple the centralised system to reduce the dependency on the communication network and on a single node to remain operations. The work has started with the introduction of an optimisation framework to integrate both controlling congestion of multiple robots and avoiding collisions. The former involves discrete variables while the other is the continuous optimisation problem. Their combination forms the mixed-integer quadratic programming (MIQP) problem where its solution on embedded devices have been introduced

recently, making the solution widely available for mobile robots. As solving the navigation in a centralised manner poses a risk as aforementioned, the distributed algorithms is investigated to solve the MIQP problem. The hypothesis is stated as:

Hypothesis 3: *The MIQP optimisation problem to address multiple path planning with congestion control and obstacle avoidance can be solved in the decentralised manner.*

Chapter 5

Thesis Results

5.1 Contributions and Results

The works done by the thesis have led to five main contributions to the researches of dependable path planning. First, the general theory about dependability has been investigated to find the fundamental properties and means to achieve a dependable path planning algorithm. Thereafter, a new path planning algorithm with dipole flow field has been developed. Third, a new application of PN on path planning algorithm to avoid congestion has been proposed. The fourth contribution is multiple path planning with its role to improve a dependable path planning with fault tolerance. Finally, the basic ideas of the thesis have been implemented in ROS to be applicable in real robots and practical application. The combination of different approaches presented in this thesis reveals the implementation of three layer protections for a robot at different ranges (Figure 5.1).

Dependable properties of a path planning algorithm For a path planning algorithm, it is crucial to enhance three dependable properties including reliability, availability, and safety. Both reliability and availability are assessed by the correctness of the path planning algorithm, i.e. robots are precisely

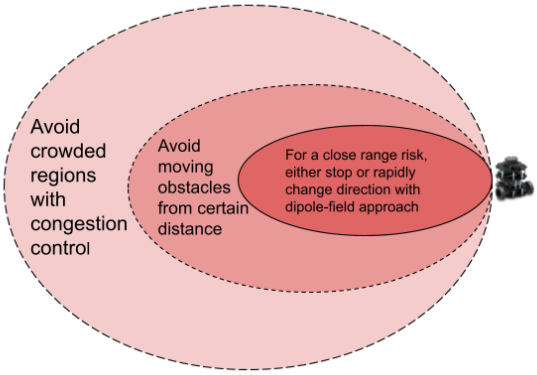


Figure 5.1: Multiple layers proposed to enhance path planning dependability.

navigated to the defined goal. Also they are shown by the continuity of path planning services, e.g. not facing a dead-lock situation so that the robots are no longer to be able to operate. Meanwhile, safety is realised by protecting robots free from collisions with any static and moving obstacles.

Dipole flow field A new path planning algorithm with dipole flow field has been developed. The algorithm is able to process path planning with a fast and effective algorithm by developing a navigation field so that the movements of agents are just simply controlled by the forces generated from the field. The attractive forces that drive the agents toward their desired goals are created by a static flow field. Simultaneously, the repulsive forces that prevent agent-agent and human-agent collisions are generated by a magnetic field of dipoles. The combination of the static flow field and dipole field forms a force to determine the moving directions of the agents at a specific time instance. The simulation results showed that the static flow field safely navigated the autonomous agents to their goals, meanwhile the dipole filed is effective in avoiding collisions. However, routing several robots into a narrow area, e.g., a corridor, might lead to a dead-live lock situation where robots try to avoid collisions

with each other but not able to escape from the trapped region. One solution to address this problem is locating potential areas that have high possibility to cause traffic jams and controlling the movements of robots through that area with the following contribution.

Congestion control with PN The application of PN is utilised to control the movement of the robots when they are routed through a cross or a narrow area. The cross/intersection regions of the moving paths are considered as a conflict resource so that a PN is an appropriate approach to avoid traffic congestion among robot. A PN is applied to control the movements of robots at every intersection by organising one by one passing through a cross. Those PN analyses and controls are available with robots when they are able to communicate and share information with each other. For humans and other moving objects, the previous work with dipole field is integrated with dynamic window approach is implemented based on the local planning of the system to help robots to avoid collisions. By regarding the velocity and direction of such dynamic obstacles as a source of a virtual magnetic dipole moments, the dipole-dipole interaction between the moving objects will generate repulsive forces to prevent collisions. Analyses and experiments demonstrated with the Gazebo simulator have revealed that the PN control is able to synchronise the movements of multiple robots passing through the intersection, which helps to shorten the travelling paths of the robots. Meanwhile, the dipole field implemented on DWA is able to advance the local path planning with an ability to avoid moving humans and other robots in the shared work-space.

Multiple path planning in both centralised and decentralised manner The next contribution is the proposed multiple path planning and its role to improve dependable path planning with fault tolerance. Relying on a single global path planning could lead to a dead - or live lock situation which happens when multiple robots moving in a narrow area and they repeat the same moving trajectories through that area again and again without finding the path to the goal. Thanks to the use of multi-paths, or back-up planned paths, the robot is able to switch to another planned path when the currently planned path

could lead to blocking. This helps to improve the availability of the system with the ability to provide correct services by proactively preventing the situation when the system fails to complete their planning tasks. The proposed multiple path planning is combined with obstacle avoidance in a unique optimisation framework. The solution for the problem is developed and presented both centralised and decentralised manner.

Implementation on ROS Lastly, one of the crucial contributions of this thesis work is the attempt to transfer the developed framework into the middleware, Robotic Operation System, which can be effectively implemented on real robots. Most of the thesis's experiments and evaluations are implemented on ROS and tested with Gazebo simulator. A basic part of the dependable path planning which is Theta* combined with dipole field, is implemented on a real HRP robot, showing that the implementation works well on a real robotic system and helps the robots avoid both static and dynamic obstacles including moving humans.

5.2 Summary of Papers

The contributions and results are detailed in four articles (Papers A-D) that are summarised as follows.

Paper A This paper presents the basic framework for dependable path planning for multiple robotic agents (or agents in short) with a field-based approach. The main navigation aim is to drive an agent to the goal with a shortest path and to avoid collisions with other agents and moving obstacles, e.g. humans. The definition of dipole flow field is proposed with the combination of static flow field and dipole field. For the static flow field, the Theta* is applied to initialise a single path from a starting point to a desired goal for each autonomous agent. Thereafter, a static flow field is configured along the found path to navigate the agents back to their planned routes if they are deviated from the path due to the meeting with static or dynamic obstacles. To deal

with collisions that may happen among moving agents, a novel dipole field is introduced. The field generates dipole forces between two moving agents (presented as two dipole magnets) approaching to each other to push them far away to avoid collision. The combination of these two fields results in a safe path planning algorithm for controlling autonomous agents to reach to their goals. Several scenarios are used to evaluate the effectiveness of the proposed approach. For a static flow field, one hundred trials of experiments have been performed to evaluate this field to drive a single agent from a start to a goal in a $50m \times 50m$ map with static obstacles. Experimental results have shown that the agent is able to move to its goals in a binary map of static obstacles with a small number of re-initialising the global path using the Theta* algorithm. The dipole field is evaluated with the simple crossing scenario of two agents moving toward each other in different orientations in an empty map. The dipole force helps them to avoid collisions yet keep them in smooth moving trajectories to their goals. With the combined dipole-flow field, the robotic agents are well routed to their destinations, while possible collisions with other agents and human are taken into account to protect them and environments from damages caused by a hit and to ensure safety to operating users. In conclusion, the basic dependable properties of completing navigation tasks to the goal and of protecting agents free from collisions have been realised with an simple yet effective approach with dipole-flow field.

Paper B In this paper, the next direction to enhance the dependability of a path planning system is investigated with the uses of Petri net (PN) for congestion analysis and movement control. Obviously, the indirect reason behind the failures of a path planning algorithm for multi-robots links to a traffic jam at a narrow area or at a crossing place. To tackle this problem, in a transportation system, the controls at each intersection, like the uses of traffic light or the moving in a queue at round-about, have been well implemented. This paper adapts those mechanisms for multiple robots to analyse and control their movements using a PN. For each robot, several paths to the goal are created and

shared among robots. A central node collects information of global paths and construct an map of intersections where each of them are crossed by at least two paths. Correspondingly, the PN is built where each robot is modelled by a token while the control place is defined a critical resource. The PN control the robot movement by allowing only a limited number of robots passing through the intersection. Other control places are also formulated at a narrow area to determine a maximum number of robot simultaneously going through it.

The whole proposed algorithm is divided into two main stages. In the first stage, the simulation of firing a token through the PN analyses how the movements of the robots look like with a specific configuration of a global path. All combinations of multi-paths assigned to robots are evaluated to find the optimal configuration of the global path for each robot. Once the global path is decided, in the second stage, the PN is utilised to synchronise the movement of a robot through the intersection in first-in-first-out manner with a link of a robot to a token in PN. The PN is mainly applied to help robots avoid mutual collisions with each other. For humans and unexpected moving obstacles, the dipole field is integrated to prevent collisions by repulsive forces generated by moving objects modelled as dipole magnets. The whole framework is implemented on an common and open platform ROS with possibility to be applicable on real robots. Analyses and experiments are demonstrated with an extensive simulator to evaluate the effectiveness of the proposed approach.

Paper C Continuously, this paper presents a simplified version of congestion control for multiple path planing. As aforementioned, a single path planning may lead to a deadlock when a robot fails to continue its moving plan. The deadlock can happen when two or more robots occupy a same place at an instance of time and the obstacle avoidance in a local range is insufficient to drive robot to another way to escape from the deadlock. Multiple-path planning plays an important key to overcome this limitation. The missing question to solve the puzzle is to how to find the right path among multiple choices to go. Therefore, this paper formulates the multiple path planning with ob-

stacle avoidance as an optimisation framework where the cost function is the summation of the path length and the minimal change in velocity and direction. The obstacle avoidance is presented by a constraint in a velocity obstacle space to select a moving direction not collide with other incoming robots. The congestion constraint is to prevent more robots moving through a narrow area. The optimisation problem is solved in a centralised manner with IBM CPLEX. Once the right path and velocity are found, they configure the moving plan of the robots with the ROS platform. The simulated experiment is conducted with three HRP robots which travel from one side of the map to the opposite one and through a narrow corridor. The starting and goal are randomly placed. The experiment has shown that the proposed optimisation has reduced the number of deadlocks and collisions, compared with conventional dynamic window approach (DWA) [69] and DWA+VO.

Paper D This paper presents a decentralised framework to tackle the path planning algorithms for multiple robots. The path planning is stated as an optimisation problem to find the optimal path to a goal for every robots in the working space to minimise or maximise a cost function. At first, the solution requires the needs of constructing multiple paths for each robot to address the situation that two or more robots occupy a same location or a same moving way at an instance of time. Correspondingly, the check of where or when a collision happens is performed to provide the constraints for the optimisation problem to avoid the conflicts. However, it is hard to determine the exact constraints of the meeting time of robots under uncontrolled environments with randomly moving obstacles and humans as the robot has to adapt its speed and moving directions to avoid them. The paper therefore proposes soft constraints which, instead of finding a conflict-free, attempt to reduce the congestion on the moving way of robots. The congestion constraints are estimated by the number of crosses or narrow areas on a specific path. By avoiding the path with heavy traffic jams, the robots proactively avoid the possibility of congestion, which could lead to a collision or a dead-locks to prevent them reaching their goals.

As allowing a change of several robots meeting each other, the obstacle avoidance is required in this situation. Several constraints are integrated into an optimisation problem with a quadratic utilisation function including an object function to minimise the travelling path and a regularisation term to make the moving path to become smooth. In overall, the whole problem is formed as a mixed integer quadratic programming (MIQP) with the integer variables to select a suitable path for each robot. The problem is NP-hard in general where its complexity grows with respect to the number of robots. Therefore, the problem is solved faster by sharing their computational resources to search for the solution of the problem. This is also to enhance the reliability of the system to reduce all dependency on one central node. The optimisation at a local robot is performed with OSQP, a light and efficient solver designed for mobile robot. The algorithms are implemented on ROS and evaluated with simulations on Turtlebot 3.

Chapter 6

Conclusions and Future Works

6.1 General Conclusions

Undoubtedly, navigation is considered as the one of the core components of an autonomous robot system. Starting from a simply requirement of routing a robot to a defined goal, nowadays the path planning problem has become increasingly challenging due to new requirements of an autonomous robot to cooperate with others and even humans to perform a task. In consequence, there have been raising questions to achieve a dependable path planning system, leading to the researches done in this thesis to find the answers for those questions. At first, a board research about dependability and its role in autonomous control has been conducted. Corresponding, three main dependable properties including reliability, availability, and safety for a path planning algorithm have been focused within the scope of this thesis. To aim at a dependable path planning system, the roots to affect those dependable properties are caused by a fault, which could happen due to operator's errors, unexpected events from surrounding environments, or even wrong designs. To

reach an acceptable level of dependability, different means have been used as the tools to improve the dependability of a system including fault prevention, fault removal, fault forecasting, and fault tolerance. From above information, the pathway towards the dependable path planning algorithm has been established to drive the researches in the rest of this work. The thesis begins with the development of a safe, reliable, and effective path planning algorithm for a group of robotic agents that share their working space with humans based on a novel dipole flow field including static flow field and dynamic dipole field. The results show that the static flow field is able to drive agents to the goals with a small number of requirements to update the path of agents. Meanwhile, the dipole field plays an important role to prevent collisions. The combination of these two fields results in a safe path planning algorithm to drive agents to their desired goals. Continuously, a fault analysis with a group of multiple robots working together to complete a shared task has been investigated using PN. Consequently, PN is applied to manage the path planning tasks of multiple robots to reduce congestion. The research of this work is continued with multiple path planning with an aim to help multiple robots to avoid the deadlock situation to ensure the availability and reliability of the path planning services. Last yet importantly, it is crucial for a dependable system to always have a reserved solution even with redundancy to avoid the completely failed of the whole system. The final attempt of this thesis work has solved the puzzle of how to bring the implementation of the whole system in a decentralised manner. The key dependable properties including reliability, availability, and safety are evaluated through extensive simulations with Gazebo. The implementation of the proposed algorithm has been transferred into ROS platform to be used in real robots. In overall, the works presented in this thesis have established a pathway and achieved important improvements to approach toward a dependable path planning system.

6.2 Future Works

Research directions have been planned for future works as follows.

Implementation on more real robots/Large scale evaluation In this thesis, the design for the dependable path planning algorithm and the theoretical analysis are developed with ROS and evaluated through Gazebo simulation. The algorithm is initially implemented on ROS-based robots, e.g. HRP. In the next stage, the implementations on other robots as well as more real world scenarios will be planned to show the effectiveness of the proposed method. Furthermore, a large scale evaluation of hundred of robots will be planned with the uses of a powerful computer for simulation.

Diverse fault tolerance and congestion control mechanisms Currently, alternative paths to the goal have been established to provide a backup path if the robot no longer follows the current path. The works can be improved by utilising different algorithms to generate global paths, like RRT-based algorithms [70] to provide more variations of the routes to the goal. A hardware solution for fault tolerance will be also considered. Meanwhile congestion control should focus more on potential areas with high probability of traffic jams instead of applying control on every intersections to reduce complexity of the system.

Hierarchy path planning The complexity of the mixed integer optimisation grows exponentially with respect to the number of discrete variables including in the problem. Therefore, the divide-and-conquer mechanism is the key to tackle the complex path planning problem involving a large number of robots in a big terrain. Robots close together in a configuration space are grouped into clusters and consequently the path planning problem is divided into the local navigation within a cluster and the global navigation among clusters through inter-connections.

Machine learning in path planning Lastly, learning techniques, such as deep reinforcement learning [71], can be investigated to make path planning decisions more accurate and fast by learning experiences from collected data.

Besides, once the system is distributed, finding the right decision will be a heavy task for each independent robot. The path planning task can be mitigated by searching information for learning/collecting data. In another aspect, applying learning methods for data fusion from extensive sensors could also help to fully perceive surrounding environment to achieve a higher level of dependable path planning for a multiple robot system.

Bibliography

[1] H. Kagermann, W. Lukas, and W. Wahlster, "Industrie 4.0: Mitdem internet der dinge auf dem weg zur 4. industriellen revolution," *VDI nachrichten*, vol. 13, no. 11, 2011.

[2] A. Avižienis, J. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, Jan-Mar 2004.

[3] I. Sommerville, *Software Engineering, 10th Edition*. Pearson, 2016.

[4] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: An open-source robot operating system," in *Workshop on open source software, IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5–11, 2009.

[5] G. Birgitand and H. Martin, "Dependable interaction with an intelligent home care robot," in *In proceedings of ICRA - Workshop of Technical Challenge for Dependable Robots in Human Environments*, pp. 21–26, 2001.

[6] M. Goran, A. Johan, and N. Christer, "A dependable real-time platform for industrial robotics," in *Proceedings of the International Conference on Software Engineering, ICSE '03*, 2003.

[7] Kalra, Nidhi, and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?," in *Research Report, Santa Monica, CA: RAND Corporation*, 2016.

[8] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," in *SAE World Congress*, 2016.

[9] A. A. ǐ zienis, J. Laprie, B. B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transaction on Dependable and Secure Computing*, vol. 1, pp. 11–33, 2004.

[10] R. Alami, R. C. S. Fleury, M. M. Ghallab, and F. Ingrand, "An architecture for autonomy," *International Journal of Robotics Research*, vol. 17, no. 4, pp. 315–337, 1998.

[11] N. Muscettola, P. P. Nayak, B. Pell, and B. C. Williams, "Remote agent: To boldly go where no AI system has gone before," *Artificial Intelligence*, vol. 103, no. 1–2, pp. 5–47, 1998.

[12] H. Bruyninckx, "Open robot control software: the orocos project," in *In IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2523–2528, 2001.

[13] D. Powell, J. Arlat, H. N. Chu, F. Ingrand, and M. O. Killijian, "Testing the input timing robustness of real-time control software for autonomous system," in *Proceedings of the European Dependable Computing Conference (EDCC)*, pp. 73–83, 2012.

[14] M. O'Brien, R. C. Arkin, D. Harrington, D. Lyons, and S. Jiang, "Automatic verification of autonomous robot missions," in *Proceedings of the International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), Springer*, pp. 462–473, 2014.

[15] B. Lussier, A. Lampe, R. Chatila, F. Ingrand, M. O. Killijian, and D. Powell, "Fault tolerant planning: towards dependable autonomous robots," in *Research Report, LAAS-CNRS*, 2015.

[16] B. Petr and G. Thomas, "A novel hazop study approach in the RAMS analysis of a therapeutic robot for disabled children," *In Computer Safety, Reliability, and Security*, pp. 15–27, 2010.

[17] J. K. Vaurio, "Fault tree analysis of phased mission systems with repairable and non-repairable components," *Reliablility Engineering and System Safety*, vol. 74, no. 2, pp. 169–180, 2001.

[18] E. Troubitsyna and K. Javed, "Towards systematic design of adaptive fault tolerance systems," in *The Sixth International Conference on Adaptive and Self-Adaptive Systems and Applications, IARIA*, 2014.

[19] S. D. Stoller and F. B. Schneider, "Automated analysis of fault-tolerance in distributed systems," *Journal of Formal Methods in System Design*, vol. 26, 2005.

[20] B. Lussier, A. Lampe, R. Chatila, F. Ingrand, M. Killijian, and D. Powell, "Fault tolerance in autonomous systems: How and how much?," in *Proceedings of the 4th IARP/IEEE-RAS/EURON Joint Workshop on Technical Challenge for Dependable Robots in Human Environments*, 2005.

[21] D. S. Yershov and S. M. LaValle, "Simplicial dijkstra and A* algorithms for optimal feedback planning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3862–3867, September 2011.

[22] T. H. Cormen, C. E. Leisercon, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*. The MIT Press, 2009.

[23] D. S. Yershov and S. M. LaValle, "Simplicial Dijkstra and A* algorithms for optimal feedback planning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (San Francisco, CA, USA), pp. 3862–3867, September 25-30 2011.

[24] S. Koenig, M. Likhachev, Y. Liu, and D. Furcy, "Incremental heuristic search in AI," *Journal of AI Magazine*, vol. 25, pp. 99–112, 2004.

[25] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A*," *Journal of Artificial Intelligence*, vol. 155, pp. 93–146, 2004.

[26] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (San Diego, CA, USA), pp. 3310–3317, May 8–13 1994.

[27] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, pp. 354–363, 2005.

[28] X. Sun, W. Yeoh, and S. Koenig, "Dynamic fringe-saving A*," in *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, (Budapest, Hungary), pp. 891–898, May 10–15 2009.

[29] H. Hu and M. Brady, "Dynamic global path planning with uncertainty for mobile robots in manufacturing," *IEEE Transactions on Robotics and Automation*, vol. 13, pp. 760–767, October 1997.

[30] K. Ok, S. Ansari, B. Gallagher, W. Sica, F. Dellaert, and M. Stilman, "Path planning with uncertainty: Voronoi uncertainty fields," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Karlsruhe, Germany), pp. 4581–4586, May 06–10 2013.

[31] Y. Wang and G. S. Chirikjian, "A new potential field method for robot path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (San Francisco, CA), pp. 977–9822, April 2000.

[32] Y. Golan, S. Edelman, A. Shapiro, and E. Rimon, "Online robot navigation using continously updated artificial temperature gradients," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1280–1287, 2017.

[33] L. A. García-Delgado, J. R. Noriega, D. Berman-Mendoza, A. L. Leal-Cruz, A. Vera-Marquina, R. Gómez-Fuentes, A. García-Juárez, A. G.

Rojas-Hernández, and I. Zaldívar-Huerta, "Repulsive function in potential field based control with algorithm for safer avoidance," *Journal of Intelligent Robot Systems*, vol. 80, pp. 59–70, October 2015.

[34] E. Owen and L. Montano, "Motion planning in dynamic environments using the velocity space," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Edmonton, Canada), pp. 2833–2838, August 2–6 2005.

[35] E. Owen and L. Montano, "A robocentric motion planner for dynamic environments using the velocity space," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Beijing, China), pp. 4368–4374, October 9–15 2006.

[36] B. Damas and J. Santos-Victor, "Avoiding moving obstacles: the forbidden velocity map," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (St. Louis, USA), pp. 4393–4398, October 11-15 2009.

[37] J. V. D. Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Pasadena, CA, USA), pp. 1928–1935, May 19–23 2008.

[38] J. V. D. Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), pp. 3475–3482, May 9–13 2011.

[39] D. Wilkie, J. V. D. Berg, and D. Manocha, "Generalized velocity obstacles," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (St. Louis, USA), pp. 5573–5578, October 11–15 2009.

[40] B. H. Lee, J. D. Jein, and J. H. Oh, "Velocity obstacles based local collision avoidance for holonomic elliptic robot," *Journal of Autonomous Robots*, vol. 41, pp. 1347–1363, 2017.

[41] A. Nash, K. Daniel, S. Koening, and A. Felner, "Theta*: Any angle path planning on grids," *Journal of Intelligent Robot System*, vol. 39, pp. 533–579, 2010.

[42] P. Yap, N. Burch, R. Holte, and J. Schaeffer, "Block A*: Database-driven search with applications in any-angle path-planning," in *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, (San Diego, CA, USA), p. 120–125, May 7-11 2011.

[43] D. Ferguson and A. Stentz, "Using interpolation to improve path planning: The field D* algorithm," *Journal of Field Robotics*, vol. 23, pp. 79–101, 2006.

[44] T. Uras and S. Koenig, "An empirical comparison of any-angle path-planning algorithms," in *Proceedings of the Symposium on Combinatorial Search (SOCS)*, pp. 206–210, 2015.

[45] S. Balu and C. Priyadharsini, "Smart traffic congestion control system," in *Proceedings of the IEEE International Conference on Computing Methodologies and Communication (ICCMC)*, (Erode, India, India), March 27–29 2019.

[46] G. Vinicius and C. Luiz, "Hierarchical congestion control for robotic swarms," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, September 2011.

[47] M. Kohlík, "Dependability models based on petri nets and markov chains," in *Information Science and Computer Engineering, 1st Class, Full-time study*, 2009.

[48] M. Malhotra and K. Trivedi, "Dependability modeling using petri-nets," *IEEE Transactions on Reliability*, vol. 44, no. 3, 1995.

[49] B. A. Bernardi, S. and S. Donatelli, "Petri nets and dependability," *LNCS Springer*, 2004.

[50] B. Samanta and B. Sarkar, "Application of petri nets for systems modeling and analysis," in *OPSEARCH*, 2012.

[51] D. Bera, *Petri nets for Modeling Robots*. PhD Dissertation, 2014.

[52] M. Thabet, A. Montebelli, and V. Kyrki, "Learning movement synchronization in multi-component robotic systems," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Stockholm, Sweden), pp. 249–256, May 16–21 2016.

[53] L. Iocchi, M. T. Lazaro, A.-I. M. Laurent Jeanpierre, and H. Sahli, "Coaches cooperative autonomous robots in complex and human populated environments," in *LNCS Springer*, 2015.

[54] G. Kim, W. Chung, S.-K. Park, and M. Kim, "Experimental research of navigation primitive selection using generalized stochastic petri nets (gspns) for a tour-guide robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Alberta, Canada), August 2–6 2005.

[55] R. Lill and F. Saglietti, "Model-based testing of cooperating robotic systems using coloured petri nets," in *ERCIM/EWICS Workshop on Dependable Embedded and Cyber-physical Systems*, 2013.

[56] W. Sheng, Q. Yang, and N. Xi, "Modeling, analysis and design for multi-robot exploration based on petri nets," in *Proceeding of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2005.

[57] L. Iocchi, L. Jeanpierre, M. T. Lazaro, and A.-I. Mouaddib, "A practical framework for robust decision-theoretic planning and execution for service robots," in *Proceeding of the 26th International Conference on Automated Planning and Scheduling*, pp. 486–494, 2016.

[58] G. Yasuda, "Discrete event behavior-based distributed architecture design for autonomous intelligent control of mobile robots with embedded petri nets," *Advances in Chaos Theory and Intelligent Control, Springer*, vol. 37, 2016.

[59] L. Iocchi, L. Jeanpierre, M. T. Lazaro, and A.-I. Mouaddib, "Personalized short-term multi-model interaction for social robots assisting users in shopping malls," in *LNCS Springer*, 2015.

[60] P. E. Miyagi and L. A. M. Riascos¡, "Modeling and analysis of fault-tolerant systems for machining operations based on petri nets," *Control Engineering Practice*, vol. 14, 2006.

[61] J.-C. Fabre, M. Lauer, M. Roy, M. Amy, W. Excoffon, and M. Stoicescu, "Towards resilient computing on ros for embedded applications," in *Proceeding of the 8th European Congress on Embedded Real Time Software and Systems (ERTS)*, 2016.

[62] J. Yu and S. M. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, pp. 1443–1449, 2013.

[63] T. S. Standley, "Finding optimal solutions to cooperative pathfinding problems," in *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pp. 173–178, 2010.

[64] G. Sharon, R. Stern, M. Goldenberg, and A. Felner, "The increasing cost tree search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 195, pp. 470–495, 2013.

[65] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[66] D. Silver, "Cooperative pathfinding," in *Proceeding of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pp. 117–122, June 01–03 2005.

[67] Z. Bnaya and A. Felner, "Conflict-oriented windowed hierarchical cooperative A*," in *Proceeding of the IEEE International Conference on Robotics and Automation (ICRA)*, (Hong Kong, China), pp. 3743–3748, 31 May–7 June 2014.

[68] J. Yu and S. M. LaValle, "Planning optimal paths for multiple robots on graphs," in *Proceeding of the IEEE International Conference on Robotics and Automation (ICRA)*, (Karlsruhe, Germany), pp. 3612–3617, May 6–10 2013.

[69] A. Filotheou, E. Tsardouliasand, A. Dimitriou, A. Symeonidis, and L. Petrou, "Quantitative and qualitative evaluation of ROS-enabled local and global planners in 2D static environments," *Journal of Intelligent and Robotic Systems*, vol. 98, pp. 567–601, 2019.

[70] S. M. LaValle, J. Kuffner, and J. James, "Randomized kinodynamic planning," *The International Journal of Robotics Research (IJRR)*, vol. 20, pp. 378–400, 2001.

[71] S. Josef and A. Degani, "Deep reinforcement learning for safe local planning of a ground vehicle in unknown rough terrain," *IEEE Robotics and Automation Letters*, vol. 5, pp. 6748 – 6755, 2020.

II

Included Papers

Chapter 7

Paper A: Toward Shared Working Space of Human and Robotics Agents Through Dipole Flow Field for Dependable Path Planning

Lan Anh Trinh, Mikael Ekström, and Baran Cürüklü
Published in Frontiers in Neurorobotics, vol. 12, 2018.

Abstract

Recent industrial developments in autonomous systems, or agents, which assume that humans and the agents share the same space or even work in close proximity, open for new challenges in robotics, especially in motion planning and control. In these settings, the control system should be able to provide these agents a reliable path following control when they are working in a group or in collaboration with one or several humans in complex and dynamic environments. In such scenarios, these agents are not only moving to reach their goals, i.e. locations, they are also aware of the movements of other entities to find a collision-free path. Thus, this paper proposes a dependable, i.e. safe, reliable and effective, path planning algorithm for a group of agents that share their working space with humans. Firstly, the method employs the Theta* algorithm to initialise the paths from a starting point to a goal for a set of agents. As Theta* algorithm is computationally heavy, it only reruns when there is a significant change of the environment. To deal with the movements of the agents, a static flow field along the configured path is defined. This field is used by the agents to navigate and reach their goals even if the planned trajectories are changed. Secondly, a dipole field is calculated to avoid the collision of agents with other agents and human subjects. In this approach, each agent is assumed to be a source of a magnetic dipole field in which the magnetic moment is aligned with the moving direction of the agent. The magnetic dipole-dipole interactions between these agents generate repulsive forces to help them to avoid collision. The effectiveness of the proposed approach has been evaluated with extensive simulations. The results show that the static flow field is able to drive agents to the goals with a small number of requirements to update the path of agents. Meanwhile, the dipole flow field plays an important role to prevent collisions. The combination of these two fields results in a safe path planning algorithm, with a deterministic outcome, to navigate agents to their desired goals.

7.1 Introduction

Until recently, robots have played a critical role in the manufacturing industry where the automatic robots perform repetitive and sometimes heavy tasks. The majority of these solutions assume high precision with respect to movements and positioning of the robots, without relying on sensors, or at least extensive sensor feedback. However, technological advancements in recent years have resulted in a shift of attention from pre-programmed automatic solutions to (semi)-autonomous systems that can operate in unstructured environments, and even co-exist with humans. As a result of this shift, robots will be more involved in our daily activities. Thus, they will be allowed to have more interactions with humans, share working space with humans as well as make their own decisions with some accepted levels of uncertain information collected from the surrounding environment. For instance, there is a rise of interest in self-driving cars where the fully autonomous mode has been investigated to help drive the car in city centers, substandard roads or busy highways without causing accidents. In the health care domain, robots are assumed to assist elderly people in their daily activities. In this context, different levels of safety need to be taken into account, e.g. develop an autonomous control to avoid executing any movements that the users do not expect and also to prevent accident caused by a person being hit by the robot. The challenges, and the opportunities, in the health care domain becomes more evident considering care at home. Going back to the main application domain, i.e. industrial robotics, it is evident that the next generation solutions assume high degree of interaction and collaboration between mixed teams of humans and robots. Obviously, the approach taken by these solutions will not exclude today's standard solutions. Thus, it is most likely that different solutions will exist side by side in the near future.

Nevertheless, the developments in autonomous robots that co-existence of humans and robots, have opened new challenges in research areas of robotics, e.g., in motion planning and control. In particular, the control system should be

able to provide the robots a reliable motion planning and control ability when the robots are working in a group or in collaboration with one or several humans in complex and dynamic environments. This means that the robots must meet certain requirements on trustworthiness/dependability in order to be allowed to work with humans. The dependability of a robotic agent is presented by main attributes including availability, i.e. the continuous operations of the system over a time interval, reliability, i.e. the ability of the system to provide correct services, and safety, i.e. the robotic agent must ensure safe controls to avoid any catastrophic consequences on users, other robots, and finally the environment. In order to implement a dependable robotic agent, important efforts have been attempted in several directions. Firstly, level of robot autonomy is automatically adaptive to the working context in order to address alternative complexities of environments. Secondly, the robot is willing to share the control with humans and other robots to optimise the working performance as well as to deal with complicated tasks that the robot cannot complete by itself. Lastly, to some extent, the robot must be able to handle the dynamic changes that occur in the environment, and to operate in accordance with the presence of other robots and humans in the same working space. This work mainly focuses on the last approach to enhance the robustness and dependability of the agents while working together with others and humans to complete a task.

Note also that, the high-level specification of a complicated movement of robots can be constructed through a sequence of lower level motion and path planning. A common problem is the movement of a robot arm, which can be composed of a sequence of trajectory planning and collision detection steps [30, 29]. Therefore, motion and path planning are concerned as the basic, and separate, constructions for plans of robotic actions. Path planning is the process which is utilised to construct a collision-free path from a starting point to a destination given a full, partial or dynamic map. Motion planning, meanwhile, is the progress in which a series of actions are needed to be defined to follow the planned path. The most common practice in robotics is to address the navigation problem using path planning, i.e. pure geometric planning from

point to point, then motion planning is to realise the feasibility of the path. As the output of path planning will later determine the way to plan robots' motion, the path planing algorithm is better incorporated with motion planning to optimise the movement of the robot. This means that the path planning could be realised at every locations within the form of navigation field to make transformation from path to motion planning easier. Besides, the moving path must be estimated to avoid many changes of moving directions to save energy used to perform movements.

With regard to above mentioned issues, this paper addresses path planning of robotic agents in the context of shared working space of humans and agents. The aim is to develop a path planning algorithm to deal with the dynamic changes of environments and complicated maps with multiple static obstacles having a wide range of shapes. The algorithm also helps agents avoid collisions with humans and others in the shared environment, in which a group of agents are designed to collaborate with each in order to plan their optimal paths, in real-time. Finally, how to combine the aforementioned factors of motion planning into the developed path planning algorithm is investigated.

So far, numerous path planning approaches have been proposed to address control movement of robots. Most of them have been focused on searching to find a path from a starting point to a destination in either static or dynamic map. Meanwhile, a family of path planning algorithms address the problem of avoiding moving obstacles with field-based approaches. Regarding search-based algorithms, one of the most conventional yet still effective approaches for the navigation of an agent in a large map is related to Dijkstra and its extension of A* searching algorithm [17, 18], and incremental search [23]. In detail, the A* algorithm improves the Dijkstra's algorithm by approximating the cost-to-go function with heuristic knowledge to reduce the searching space to the goal. Meanwhile, incremental search algorithms seek for the shortest paths by utilising the results of similar searches to make the search faster, instead of solving each search problem separately. By applying incremental search on top of the A*, [19] developed lifelong planning A* (LPA*) as an initial variant of

A*, in order to address path planning for dynamic graphs with changing edge costs. In the D* algorithm [24], incremental search is applied to repeatedly update the shortest paths between the current position of a robot and a goal, during the robot's approach to the goal. Koenig and Likhachev [20] improved the D* by LPA* and alternatively Sun et al. [22] developed dynamic fringe saving A* to reuse the OPEN and CLOSED lists from previous A* searches. Although different variants of A* are able to address a graph change due to the moving of a robot to a new vertex, or the updates of edge costs, those algorithms still face difficulties to deal with moving obstacles. In addition, as stated by Hu and Brady [1], a probabilistic approach is necessary to model the uncertainties of mobile obstacles in the environment. However, the complexity of path planning will be significantly increased if either the cost of the edges, or the links of the graph are presented by random variables.

In order to handle the uncertainties of observed obstacles, a field-based approach is another way to find the path for the agents. The field is calculated for each location, in time and space, and determines the directions of movement of an agent to reach the destination. The field consists of a repulsive field to push the agent away from the obstacles, and an attractive field to pull the agent towards the goal. For instance, Ok et al. [3] proposed Voronoi uncertainty field which is build from Voronoi diagram from the start to the goal to create the attractive field and the repulsive field from the robot to the obstacles. The works of Wang and Chirikjian [5] and later Golan et al. [4] presented an artificial potential field based on the exchanges of heat flow. If obstacles are visualised as hot objects, the target is then presented as the cold one and the temperature is discretised at each location on the grid. The temperature gradient solved by partial differential equation generates the appropriate forces to drive the robot. One of the big issues of using the potential field is that the repulsive field may push the agent to reach other obstacles or statures with the attractive field. Due to these problems, the agent may be trapped into a local optimum or loose its way toward the goal. To mitigate the local converge to a local optimal, some additions to the potential field have been introduced. Valbuena and Tanner [7]

proposed the way of adding velocity constraints, meanwhile García-Delgado et al. [8] extended the repulsive function with the change of magnitude dependent on the angle between the attractive force and the obstacle. The main aim is to avoid the cancellation of the repulsive and attractive forces when applied in opposite orientations. However, the interactions of the agents with the environment, especially changes in the map, were not clearly addressed in above mentioned works. Besides, most of field-based navigation approaches lack the global information of a feasible path to the destination that could actually help avoid a trap that would lead to a local optimum.

Controlling the speed and directions of a robot are also key factors, which plays a role to provide the robot a collision free path. Owen and Montano [10, 11] defined velocity space to estimate the arrival time of moving objects to a region of potential collisions and thereby potential solutions to avoid these collisions. The velocity space in which the motion of the robot, as well as static and moving objects are mapped, is applied to predict when the collision may happen and when the robot may escape from the collision. Damas and Santos-Victor [12] developed a map of forbidden velocity zones which is constructed as a limit on the velocity of the robot to avoid collision with obstacles. When the robot moves into the forbidden zones, it may adjust its speed to avoid the obstacles. Berg et al. [9] Wilkie et al. [14], and Berg et al. [13] further integrated the acceleration while Lee et al. [15] concerned the shape of the robots as an ellipse for obstacle avoidance. Yoo and Kim [16] proposed a modified uni-vector field to present obstacles with respect to relative their velocities and positions where the gaze control which concerned the error of localisation and the distances to surrounding obstacles was also combined into the system to find the best moving trajectory. Belkhouche [2] introduced virtual plane to present moving objects with information of velocity into stationary ones. As a consequence, path finding in a dynamic environments is converted to a simpler problem of navigation in a static environment. However, it is noted that, it is not always optimal to use velocity planning when to drive the robot. Using only velocity control for path planning usually results in oscillatory motion. Given

a typical differential drive mobile robot, there are a number of constraints on the linear and angular velocities, as well as the acceleration, in order to save energy for extending operation time, and finding the path to the goals with few turns. To the best of our knowledge, these concerns have not been investigated extensively in combination with obstacle avoidance in dynamic environment.

In order to address the above mentioned issues, in this paper, a novel method for path planning of mobile agents, in the shared working environment of human and agents, called as the dipole flow field, is proposed. The dipole flow field combines both global and local path planning in a unique framework. For global planning, the method applies any-angle path planning algorithm of Theta* [6] to generate smooth paths with few turns, from a starting point to a goal for a pool of agents. Although different A* variants of any-angle path planning haven been proposed, such as A* post smoothing, block A* [25] and field D* [21], the Theta* is able to provide the most optimal path with simple and effective implementation [26]. As the computations of the Theta* algorithm is costly for a big map, the algorithm is updated when there is a significant change on the static map of the environment. To cope with dynamic movements of the agents, a static flow field along the planned path is defined to attract the agent back to continue reaching the goal even when the agents may be deviated from the planned path. In addition, a dipole field is used to avoid the collision of the agents with others and human within shared working space. To the best of our knowledge, most conventional approaches attempt to generate the pushing forces based only on the location of the agents, whereas in this work, it is assumed that, those should be better aligned with both moving directions and velocity magnitudes of different agents. The generated dipole field is able to push other agents far away based on their respective moving directions and the velocity magnitude of the agents. Static flow field and dipole field are combined to assure a dependable path of each agent from the starting point to the goal without colliding with each other.

The rest of paper is organised as follows. The methodology of the proposed path planning based on dipole flow field is presented in Section 2. The evalu-

ation of the proposed method through the experimental results is described in Section 3. Finally, the paper is concluded with discussion in Section 4.

7.2 Methodology

In this section the agent architecture together with the different modules used for path planning are described in Section 7.2.1. Meanwhile the core path planning algorithm is presented in Section 7.2.2.

7.2.1 Autonomous agent architecture

The overall architecture of the autonomous agent to support the proposed planning algorithm is presented in Figure 7.1.

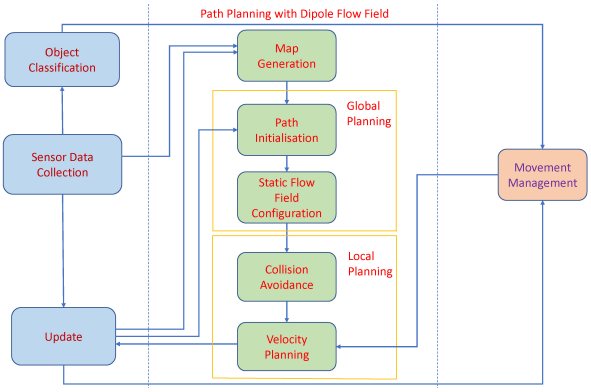


Figure 7.1: The Architecture of Autonomous agent. The backbone of the path planning algorithm consists of the *Map Generation* module, *Global Planning* including the *Path Initialisation* module and the *Static Flow Field Configuration* module, and *Local Planning* including the *Collision Avoidance* module and the *Velocity Planning* module.

The core algorithm includes the following five modules, *Map Generation, Path Initialisation, Static Flow Field Configuration, Collision Avoidance* and

Velocity Planning. In addition, there are four external modules including *Sensor Data Collection, Update, Object Classification*, and *Movement Management* to help the planning algorithm collect information from the surrounding environment and update control.

Path planning architecture

After that the global information of the environment is acquired from the external modules, a 2-D map is generated. The 2-D map is presented as a binary map in which static objects and obstacles are shown as black areas whereas the allowed moving areas are illustrated with white colour. Global path planning with Theta* algorithm is applied in the *Path Initialisation* module to initiate a path from the starting point to the destination for the agents. While moving to the goal, the agent may deviate from the original path due to obstacle avoidance, or accumulated errors related to velocity and pose estimating. As a result, a static flow field generated in the *Static Flow Field Configuration* module will drive the agent back to the designed path. Only when the agent moves far away from the region covered by the static flow field, Theta* is activated to renew the path from the current position to the goal. After the static flow field is configured, the agents start moving to reach their individual goals, while checking for collision with other moving objects. The dipole field is calculated in the *Collision Avoidance* module to avoid collision with the agents. Finally, the motions of the agents are controlled by the superposition of the static flow field, and dynamic dipole field to generate the dipole-flow force. The dipole-flow force is presented by the adjustments of the agents' heading angles. A velocity function is established to help the agent well adapt its moving velocity according to two factors, energy consumption and obstacle avoidance. If there is no collision, the agent will move with a stable speed along the configured path. Meanwhile, if there is a dynamic obstacle, the agent needs to adjust its moving direction to avoid the obstacle while still maintaining, or at least minimising, the deviation from the time to goal.

External modules to support path planning algorithm

The *Sensor Data Collection* module is designed to continuously collect information of the environment. For instance, the visual data obtained from a camera, together with the data from the other sensors, is used to build the map of the environment and to recognise different objects. The pose of the robot is collected from the inertial measurement unit (IMU). Similarly, the positioning tracking system registers the position of the robot within the map. The *Object Classification* module receives the data from the *Sensor Data Collection* module to determine which objects in the environment that are static objects and which ones are moving objects. In this work, the proposed path planning algorithm deals with two types of moving objects. Firstly, autonomous agents, which share information about their locations, and velocities, with the other agents. Secondly, uncontrolled moving object, e.g. a human, who can suddenly appear in the working space of the agents. Especially when the human subjects are present, the agents need to adjust their movement to avoid them. The *Movement Management* module plays a central role in managing the location, and moving trajectories of all agents and human(s) found in the environment. The data from the *Movement Management* module is sent to the path planning algorithm for velocity estimation. The *Update* module updates the internal model based on the changes in the environment, and applies the control commands from the *Velocity Planning* module to move the agents accordingly.

7.2.2 Path planning with dipole flow field

In this section, the dipole flow field is firstly formulated by the combination of the static flow field and the dynamic dipole field. Later, the direction of the dipole flow field at every point is turned into velocity planning to control the linear and angular velocities of agents.

Static flow field for global path planning

The global path consists of a sequence of line segments from the start to the goal, and is configured using the Theta* algorithm. Within the neighbourhood of the found path, a navigation field parallel to the path segment, is defined, as the static flow field.

Path Initialisation To initialise the path from a starting point to an ending point, the Theta* algorithm is applied. This algorithm improves A* by adding a line-of-sight (LOS) detection to each search iteration to find a less zigzaggy path than those generated by A* and its other variants. The primary difference between the Theta* and the others is that the Theta* algorithm determines the parent of a node to be any node in the searching space. Thence, the LOS detection feature is purposed to help decrease the undesired expanding by checking for whether the offspring node and the parent are in a straight line, i.e. line-of-sight. By this means, the path found by Theta* is not a connection of adjacent nodes but a connection of line-of-sight ones. The pseudo codes of the Theta* is described in Algorithm 1.

As a heuristic-based search algorithm, Theta* approximates the length of the shortest path based on cost evaluation. The cost evaluation is conducted from the f-value, i.e. the lowest cost from the starting node to the last node, s, in a path, referred to as $f(s)$, and a heuristic value called h-value which is the cost estimation from the node s the goal. The estimated cost of the cheapest node through node s is, thus expressed by

$$f(s') = f(s) + h(s, s'). \tag{7.1}$$

In this work, the heuristic function is simply defined as the Euclidean distance i.e., $h(s, s') = w.Euclidean(s, s')$ where w is a weight that determines the size of the area to search for the optimal path around the straight-line between s and s'. With $w > 1$, Theta* is able to reduce the searching area but may return a longer path, therefore the value $w = 1$ is used in this work to search for the

Algorithm 1 Theta* algorithm

procedure THETA*(s_{start}, s_{goal}) ▷ Find the shortest path from start to goal location
  $open \leftarrow \emptyset$, $closed \leftarrow \emptyset$,$g[s_{start}] \leftarrow 0$, $parent[s_{start}] \leftarrow s_{start}$
  $open.insert(s_{start}, g[s_{start}] + h[s_{start}])$
  while $open \neq \emptyset$ **do**
    $s \leftarrow open.pop()$
    if $s = s_{goal}$ **then**
      return *"path found"* ▷ The found path is stored in $parent[]$
    end if
    $closed \leftarrow closed \cup \{s\}$
    for $s' \in neighbor(s)$ **do**
      if $s' \notin closed$ **then**
        $g[s'] \leftarrow \infty$, $parent[s'] \leftarrow NULL$
      end if
      $g_{old} \leftarrow g[s']$
      COSTEVALUATION(s, s')
      if $g[s'] < g_{old}$ **then**
        if $s' \in open$ **then**
          $open.remove(s')$
        end if
        $open.insert(s', g[s'] + h[s'])$
      end if
    end for
  end while
  return *"no path found"*
end procedure

procedure COSTEVALUATION(s, s')
  if $LOS(parent[s], s')$ **then** ▷ LOS check between $parent[s]$ and s'
    if $f(parent[s]) + h(parent[s], s') < f[s']$ **then**
      $parent[s'] \leftarrow parent[s]$
      $f[s'] \leftarrow f(parent[s] + h(parent[s], s')$
    end if
  else
    if $f[s] + h(s, s') < f[s']$ **then**
      $parent[s'] \leftarrow s$
      $f[s'] \leftarrow f[s] + h(s, s')$
    end if
  end if
end procedure

shortest path. It is assumed that the straight line distance between two nodes would never be longer than any other path connecting them. However, the shortest path generated by the A* algorithm is connected by the neighbouring grid nodes, and thus entails many turning points to the robot. The path found by Theta* is a sequence of LOS nodes so that it is smoother with few turns and closer to a straight-line path between the start and the goal. The algorithm for LOS function is implemented with a drawing-line algorithm in graphics to optimise processing time and is referred to approach proposed by Nash et.al. [6].

As mentioned in Section 7.2.1, the input to the Theta* algorithm is the binary map of the environment (Figure 7.2A). However, to avoid searching the path on a dense graph, a grid-based graph is used (as visualised in Figure 7.2B). The obstacle areas are also dilated corresponding to the size of agents so that the path found by Theta* will not cause the boundary of the agent touching the edges of the map while the agent is moving.

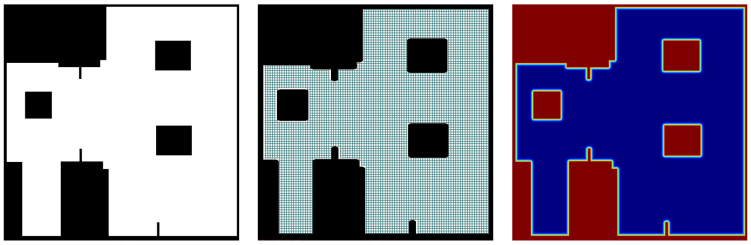


Figure 7.2: Binary map for static flow field and derived information, (A) the original binary map in which the white pixel presents available regions of agents, (B) the grid-based graph derived from the binary map, and (C) the corresponding repulsive field in which the amplitude of the field from the lowest to the highest is mapped into colors from blue to red respectively.

Path Configuration with Static Flow Field Searching for a global path from a start to a goal in a big map is a computationally heavy task, thus it is not

desired to re-calculate the path for small updates of the map, or small deviations from the configured path. The static flow field is to draw the agents back to their moving paths in those situations. In the form of force interaction, the static flow field is also easily combined with a dynamic field for obstacle avoidance. As the shortest path found by Theta* is the connection of several line segments, the static flow field is created within the neighbour of the line segments. For each path, it is assumed that there are n line segments from the start to the end points. Each line segment i is presented in a vector form of $\mathbf{x}(t) = \mathbf{a}_i + t\mathbf{n}_i$ where $\mathbf{a}_i$ is the starting of the line segment and $\mathbf{n}_i$ is the unit vector of the line. To ensure that the static flow field will draw the agent to its goal, those line segments also includes the last line segment with $\mathbf{a}_i$ is set to the goal and $\mathbf{n}_i$ to a zero vector. The flow field force at the point $\mathbf{p}$ close to the provided path found by Theta* is calculated by

$$F_{flow}(\mathbf{p}) = F_a(\mathbf{p}) + F_r(\mathbf{p}) \tag{7.2}$$

where $F_a(\mathbf{p})$ is the attractive force to draw agent back to the configured path, and $F_r(\mathbf{p})$ is the repulsive force from nearby static obstacles. The configuration of the global path and the formulation of the flow force are described in Figure 7.3.

Let $F_{a_i}(\mathbf{p})$ be the attractive force of a point to each line segment and expressed by

$$F_{a_i}(\mathbf{p}) = (1 - \mathrm{e}^{-k_1 d(\mathbf{p},\mathbf{a}_i)})((\mathbf{a}_i - \mathbf{p}) - ((\mathbf{a}_i - \mathbf{p}) \cdot \mathbf{n}_i)\mathbf{n}_i) + k_2 \mathrm{e}^{-k_1 d(\mathbf{p},\mathbf{a}_i)}\mathbf{n}_i \tag{7.3}$$

where $d(\mathbf{p}, \mathbf{a}_i)$ is the distance from the point $\mathbf{p}$ to the line segment i-th, k_1, k_2 are constants, and " $\cdot$ " denotes the inner product of two vectors. As $\mathbf{n}_i$ is a unit vector, the vector $(\mathbf{a}_i - \mathbf{p}) - ((\mathbf{a}_i - \mathbf{p}) \cdot \mathbf{n}_i)\mathbf{n}_i$ is normalized before (7.3) is calculated. Two constants $k_1 = 0.01$ and $k_2 = 1$ are selected to control the impact of the first and second terms of equation (7.3). The attractive force $F_a(\mathbf{p})$ is set equal to the attractive force $F_{\mathbf{a}_i^*}$ of the line segment $\mathbf{a}_i^*$

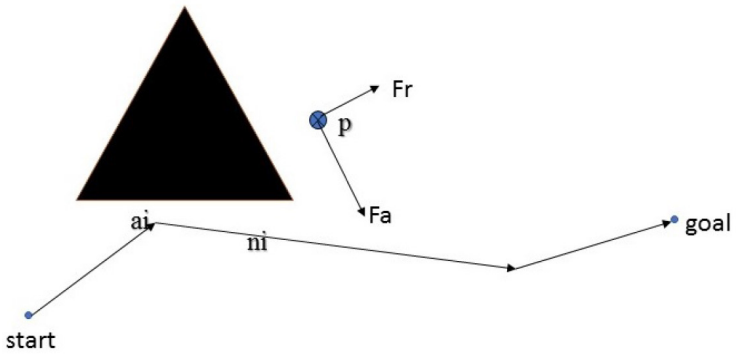


Figure 7.3: The configuration of the global path.

closest to the point $\mathbf{p}$, $\mathbf{a}_i^* = \underset{\mathbf{a}_i}{\operatorname{argmin}} d(\mathbf{p}, \mathbf{a}_i)$. Meanwhile, the repulsive force $F_r(\mathbf{p}) = -\nabla U_{rep}(\mathbf{p})$ is the negative gradient of the repulsive field

$$U_{rep}(\mathbf{p}) = \begin{cases} \eta\left(\frac{1}{d(\mathbf{p},\mathbf{p}_0)} - \frac{1}{d_0}\right)^2 & d(\mathbf{p},\mathbf{p}_0) \leq d_0 \\ 0 & d(\mathbf{p},\mathbf{p}_0) > d_0 \end{cases} \tag{7.4}$$

in which $d(\mathbf{p}, \mathbf{p}_0) = \|\mathbf{p} - \mathbf{p}_0\|$ is the Euclidean distance from the agent's position $\mathbf{p}$ to the closest obstacle's position $\mathbf{p}_0$, d_0 is the influence distance of the force, and η is a positive constant. To avoid singularities of equation (7.4) when $d(\mathbf{p}, \mathbf{p}_0) = 0$, a linear transformation $f(d) = \kappa d + 1$ is applied to map $d(\mathbf{p}, \mathbf{p}_0)$ and d_0 to non-zero values $f(d(\mathbf{p}, \mathbf{p}_0))$ and $f(d_0)$ in equation (7.4). The influence distance of the force, d_0, is selected based on the size of agents (the diameter of agents) to prevent touching the agents to static obstacles. With respect to $0 < d_0 < 100$, $\kappa = 0.01$ and $\eta = 10^4$ are chosen. An example of repulsive field of the binary map given in Figure 7.2A,B is shown in Figure 7.2C. The static flow fields without and with added repulsive forces are presented in Figure 7.4A and Figure 7.4B respectively.

The affecting area of the static flow field is determined by the window size

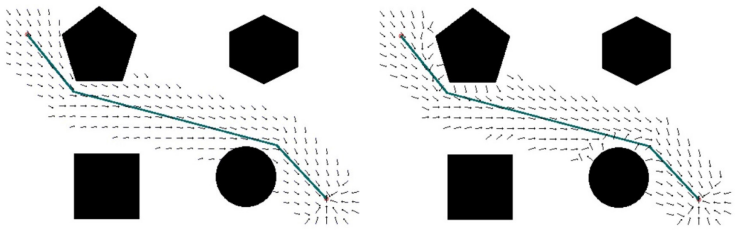


Figure 7.4: The representation of the static flow field (unity vectors), (A) the initial path with the configured static attractive field, (B) the static flow field with added repulsive force to the obstacles.

(W). This means that the static flow field remains influence on the agent if the distance from the agent to the designed path is less than W. Once the agent moves out of the affecting area, the Theta* algorithm needs to be recalculated to update a new static flow field.

Dynamic dipole field

To cope with the problem of collision avoidance, the dipole field for each dynamic object is generated. The development of the dipole field is inspired by the way that humans naturally avoid moving obstacles: When facing an obstacle that is approaching, the human may turn, and continue to move, to avoid the obstacle instead of going backwards. Such a movement shows a moving trajectory similar to that of a dipole magnetic field line. This method is also a more skillful obstacle avoidance strategy than the conventional method of using radial potential field. Munasinghe et.al [31] introduced an implementation of this obstacle avoidance method by designing a force to drive a robot through an elliptical trajectory to go around and then behind obstacles. In the work of Igarashi et.al. [32] the dipole characteristics is expressed as a vector field to push an object to a goal. In this work, to model the moving behaviour of agents, instead of developing dipole-like vector field the theory of dipole mag-

netic field in physics is directly applied. Each agent can be seen as a source of a magnetic dipole field, in which the magnetic moment is proportional to the velocity vector of the agent. This means that the orientation of the moment is aligned with the moving direction of the agent and the magnitude of the moment is equal to the speed of the agent. The aim of having the moment proportional to the speed is to ensure that among different obstacles having the same distance to the agent, the one with the larger speed will contribute a stronger effect on driving the agent. In physics, the magnetic field $\mathbf{M}$ of the dipole moment vector $\mathbf{m}$ is expressed by

$$\mathbf{M}(\mathbf{m}, \mathbf{d}) = \rho(3(\mathbf{m} \cdot \hat{\mathbf{d}})\hat{\mathbf{d}} - \mathbf{m})/d^3 \tag{7.5}$$

where $\mathbf{d}$ is the distance vector, $d = \|\mathbf{d}\|$ is distance between two agents, and $\hat{\mathbf{d}} = \mathbf{d}/\|\mathbf{d}\|$ is a unit vector. The magnetic constant $\rho = 1/3$ ($3\rho = 1$) is applied in this work instead of using $\rho = \frac{\mu_0}{4\pi}$ in electromagnetic theory (μ_0 is the permeability of free space). An agent with the magnetic moment $\mathbf{m_j}$ within the magnetic field $\mathbf{M_k}$ generated by the other magnetic source $\mathbf{m_k}$ would be affected by the force

$$\mathbf{F} = \nabla \mathbf{m_j} \cdot \mathbf{M_k} \tag{7.6}$$

where the gradient ∇ presents the changes of the quantity $\mathbf{m_j} \cdot \mathbf{M_k}$ per unit distance. Hereby, the repulsive force of an agent k on an agent j can be formulated by,

$$\begin{aligned}
\mathbf{F}_{dipole}(\mathbf{m}_j, \mathbf{m}_k, \mathbf{d}) &= \rho\nabla\Big(\mathbf{m_j} \cdot \frac{3(\mathbf{m_k} \cdot \hat{\mathbf{d}})\hat{\mathbf{d}} - \mathbf{m_k}}{d^3}\Big) \\
&= \rho\nabla\Big(\frac{3(\mathbf{m_j} \cdot \mathbf{d})(\mathbf{m_k} \cdot \mathbf{d})}{d^5} - \frac{(\mathbf{m_j} \cdot \mathbf{m_k})}{d^3}\Big) \\
&= \rho\Big(3(\mathbf{m_j} \cdot \mathbf{d})(\mathbf{m_k} \cdot \mathbf{d})\nabla\frac{1}{d^5} + \frac{3(\mathbf{m_j} \cdot \mathbf{d})}{d^5}\nabla(\mathbf{m_k} \cdot \mathbf{d}) + \\
&\quad \frac{3(\mathbf{m_k} \cdot \mathbf{d})}{d^5}\nabla(\mathbf{m_j} \cdot \mathbf{d}) - (\mathbf{m_j} \cdot \mathbf{m_k})\nabla\frac{1}{d^3}\Big) \\
&= \frac{3\rho}{d^4}\Big((\mathbf{m}_j \cdot \hat{\mathbf{d}})\mathbf{m}_k + (\mathbf{m}_k \cdot \hat{\mathbf{d}})\mathbf{m}_j + (\mathbf{m}_j \cdot \mathbf{m}_k)\hat{\mathbf{d}} - 5(\mathbf{m}_j \cdot \hat{\mathbf{d}})(\mathbf{m}_k \cdot \hat{\mathbf{d}})\hat{\mathbf{d}}\Big)
\end{aligned} \tag{7.7}$$

where $\mathbf{m}_j$, and $\mathbf{m}_k$ are the dipole moments of the agents. To lead to equation (7.7), the gradients of two functions, $\nabla\frac{1}{d^n} = -n\frac{\mathbf{d}}{d^{n+2}}$ and $\nabla(\mathbf{m} \cdot \mathbf{d}) = \mathbf{m}$, are used.

The magnetic force $\mathbf{F}_{dipole}(\mathbf{m}_j, \mathbf{m}_k, \mathbf{d})$ is aligned with the direction of from $\mathbf{m_k}$ to $\mathbf{m_j}$ to generate repulsive forces. This means $\mathbf{F}_{dipole}(\mathbf{m}_j, \mathbf{m}_k, \mathbf{d})$ will be reversed if it has an opposite direction of a vector pointing from an agent k to an agent j. In order to increase the interaction range of dipole field, an adjustment factor γ, $0 < \gamma \leq 1$ and close to one ($\gamma \approx 1$), is added as follows,

$$\begin{aligned}
\mathbf{F}_{dipole}(\mathbf{m}_j, \mathbf{m}_k, \mathbf{d}) = \frac{3\rho}{d^{4\gamma}}\Big((\mathbf{m}_j \cdot \hat{\mathbf{d}})\mathbf{m}_k + (\mathbf{m}_k \cdot \hat{\mathbf{d}})\mathbf{m}_j + (\mathbf{m}_j \cdot \mathbf{m}_k)\hat{\mathbf{d}} - \\
5(\mathbf{m}_j \cdot \hat{\mathbf{d}})(\mathbf{m}_k \cdot \hat{\mathbf{d}})\hat{\mathbf{d}}\Big).
\end{aligned} \tag{7.8}$$

The smaller value γ is, the further distance the dipole field of one agent has influence on the others. In addition, a small term $\epsilon = 10^{-12}$ is added into d in the denominator of equation (7.8) to avoid singularities.

Dipole flow field

An agent needs to adjust its moving path according to its relative locations and orientations to other agents. Also, the agent concerns the possible collisions with uncontrolled moving objects i.e, humans, which does not share information about their locations, and intentions regarding how they will move. Assume that there are a set of N agents, i.e. robots $\mathcal{A} = \{j | j \in 1, 2, ..., N\}$ in the working space. All agents are designed using the same architecture to cooperate with each other to plan global movements so that each of them transmits location information to the other agents in $\mathcal{A}$. Let $\mathcal{O}_j = \{o_j | o_j \in 1, 2, ..., N_j\}$ be a set of N_j human subjects recognised by the agent j within its detecting range. In this context, the relative location and velocity information about human subjects are estimations from observations over time. The dipole flow field for an agent j is formulated by integration of the static flow field, and the dynamic dipole field as

$$\mathbf{F}_{df}^{(j)} = \alpha \mathbf{F}_{flow}^{(j)} / \|\mathbf{F}_{flow}^{(j)}\| + \beta_A \sum_{k \in \mathcal{A}, k \neq j} \mathbf{F}_{dipole}(\mathbf{m}_j, \mathbf{m}_k, \mathbf{d}_{jk}) + \beta_O \sum_{l \in \mathcal{O}_j} \mathbf{F}_{dipole}(\mathbf{m}_j, \mathbf{m}_l, \mathbf{d}_{jl}) \quad (7.9)$$

where $||\mathbf{F}_{flow}^{(j)}|| = \sqrt{(\mathbf{F}_{flow}^{(j)_x})^2 + (\mathbf{F}_{flow}^{(j)_y})^2}$ is the magnitude of the flow force $\mathbf{F}_{flow}^{(j)} = [F_{flow}^{(j)_x}, F_{flow}^{(j)_y}]^T$, here α, β_A, and β_O are constants. Those constants determine the impact of dipole flow forces over static flow forces to control the moving of agents. Since the static flow force is normalised in equation (7.9), the coefficient $\alpha > 0$ represents for the magnitude of the static flow field term. To simply reflect the effective area of the static flow field, $\alpha = 10$ is chosen (correspondent to the agents' diameter of $1m$, or 10 pixels, in all experiments). Meanwhile, the dipole field coefficients, β_A and β_O, determine the effecting area of the dipole field. It is able to define this area of influence of the agent (k) on (j) by a circle $\mathcal{C}_{jk}$ that has a center at the agent (k) and a radius r_{jk} to ensure that if $d_{jk} < r_{jk}$ then $\beta_A ||\mathbf{F}_{dipole}(\mathbf{m}_j, \mathbf{m}_k, \mathbf{d}_{jk})|| > \alpha$. As the magnitude of

$||\mathbf{F}_{dipole}(\mathbf{m}_j, \mathbf{m}_k, \mathbf{d}_{jk})||$ is proportional to $1/d_{jk}^{4\gamma}$, the dipole forces has strong influence on the agent (j) when the agent is inside $\mathcal{C}_{jk}$. On the contrary this influence is significantly decreased outside $\mathcal{C}_{jk}$. In this work, the two constants β_A and β_O are set to be equal ($\beta_A = \beta_O$) and defined to control the desired effective area of the dipole field. This area has a radius that is proportional to $(\beta_A/\alpha)^{-1/4\gamma}$ and to the magnitude of dipole moments of agents. It is also noted that two agents (j) and (k) receives the dipole forces with the same amplitude but with opposite directions. Only agents are affected by the dipole forces generated by human subjects. Thus, in the model human subjects are not subject to these forces.

Velocity planning

In this work, an autonomous agent is presented by the kinematics model of a unicycle-type mobile robot [34]. This model is chosen because despite its unicycle name, it approximates many widely used differential drive robots and can be easily extended to car-like mobile robots with two parallel driven wheels. The state of a robot (Figure 7.5) is described by a set of triple parameters $\mathbf{s}(t) = [x(t), y(t), \theta(t)]^T$, and $\mathbf{r}(t) = [x(t), y(t)]^T$ are the coordinates, $\theta(t)$ is the orientation with respect to the $x-$axis of the robot, and t is time. The state $s(t)$ is updated for every interval Δt as

$$\begin{aligned} x(t+\Delta t) &= x(t) + u(t)\Delta t \cos\theta(t) \\ y(t+\Delta t) &= y(t) + u(t)\Delta t \sin\theta(t) \\ \theta(t+\Delta t) &= \theta(t) + \omega(t)\Delta t \end{aligned} \tag{7.10}$$

where $u(t)$ and $\omega(t)$ are the linear and angular velocities of the agent respectively. Those velocities are computed by the following equation

$$u(t) = k_u \tanh(||\mathbf{r}(t) - \mathbf{r}_{goal}||)$$
$$\omega(t) = -k_\omega \left(\theta(t) - \arctan\left(\frac{F_{df}^y}{F_{df}^x} \right) \right), \tag{7.11}$$

where $k_u > 0$ and $k_\omega > 0$ are two constant control gains. From this definition, the linear velocity $u(t)$ is about k_u while an agent is moving on its ways and decays to zero when it is closer to the goal. Therefore, k_u is set to the expected speed of agent. Meanwhile, the angular velocity $\omega(t)$ is used to adjust the heading angle $\theta(t)$ of the agent to make the agent's orientation aligned with the direction of dipole field force $\mathbf{F}_{df}$. By this, the dipole flow field mainly affects the angular velocity $\omega(t)$ of the agent to drive it to the goal and to avoid the static obstacles, and moving objects when they are close. The second coefficient, k_ω, controls how smooth the moving trajectory of the agent is and how fast the agent is able to adapt to the changes of the dipole field force.

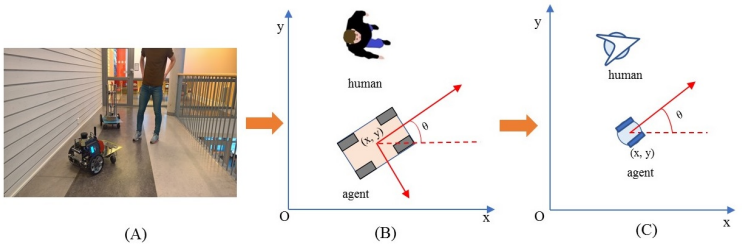


Figure 7.5: Visualisation of an agent with kinematic parameters and human from (A) a real world space in (B) a 2D mapping space, and (C) a simplified visualization used in the proposed work.

7.3 Experiments

A number of experiments are conducted to validate the effectiveness of the proposed path planning algorithm. Different with most of existing approaches which have focused on alternative aspects of local or global path planning for a single agent, this work has developed a new promising framework to address the navigation problems of multiple agents sharing working space with human. This also adds a new dimension to existing solutions of robotics navigation with the definition of dynamic dipole field inspired from electromagnetic physics and of the static flow field based on Theta* algorithm. Thus, the main aim of this section is to investigate on the characteristics of the proposed approach through various scenarios. The starting point is an experiment with static flow field. This experiment shows how this field is able to navigate agents to goals within the map of complicated static obstacles. The next experiment exploits the benefit of dipole field to help agents avoid moving obstacles coming from different directions. Finally, a set of experiments are conducted in order to evaluate how well the proposed path planning algorithm with the combination of flow and dipole flow fields, i.e. dipole-flow field, both drive agents toward the goals, and at the same time avoid collisions with moving objects. Data showing the agent-agent and human-agent distances in the presence of the dipole-flow field is also shown as part of the last experiment.

7.3.1 Static flow field

The aim of the static flow field is to convert the path found by Theta* into a navigation field to avoid the needs of running Theta* for every update of the agent position, and also to allow a more robust integration of the path planning with obstacle avoidance and velocity controls. Thus, only when the agent deviates from its designated path, due to slow adaptation to follow the navigation field, the path is required to be renewed using the Theta* algorithm. Different examples of agent movements with static flow field are shown (Figure 7.6). In most situations, like examples given in Figure 7.6A and Figure 7.6B, the agent

approaches the goals without the needs of renewing the shortest path to the goal. However, in a particular case where the agent deviates from the designed path, Theta* is reused to update the path to the goal (Figure 7.6C).

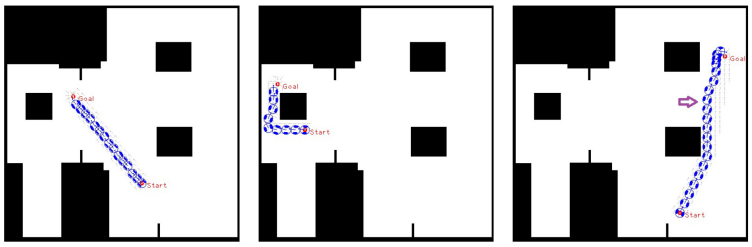


Figure 7.6: Agent moves from start to goal with static flow field where the window of static flow field is set as two times as the size of the agent. (A) and (B) an agent approaches the goal without the needs of re-estimating a new path, and (C) Theta* is reactivated when the agent gets close to the second obstacle along its path (the location for activation is shown by the arrow symbol).

Different windows of the static flow field are evaluated. One hundred trials are attempted for each specific value of the window. In this experiment, a binary map of 50 x 50 m with a resolution of 10 pixels per meter is used. Each agent is presented by a bounding circle with a radius of 0.5 m and has the speed of $0.5m/s$ with $k_\omega = 1.2$ (k_u is set to the speed of agents in all experiments). The influence distance d_0 is set to 10 pixels (or one meter). For each trial, an agent moves from a starting point to a goal using only static flow field with velocity control. Pairs of starting and ending locations are selected randomly in the map. The results reveal that the bigger window is, the less number of running Theta* the static flow field needs (Table 7.1). For the following experiments in Section 7.3.2 and Section 7.3.3, the window of two times of the agent size ($W = 2S$) is applied.

Table 7.1: Relationship between an average number of Theta* used for static flow field to successfully drive an agent to its goal and window size (W). (The size of an agent is S)

Window size	Average number of Theta*
$W = S/4$	10.85
$W = S/2$	4.57
$W = S$	1.53
$W = 3S/2$	0.73
$W = 2S$	0.43
$W = 5S/2$	0.23

7.3.2 Dipole flow field for crossing scenarios of two agents

To analyse the behaviour of dipole flow field for obstacle avoidance, two simple scenarios, in which two agents are crossing each other are chosen (Figure 7.7). In Scenario 1, one of the agent moves from left to right and the other agent moves in the opposite direction. In Scenario 2, the first agent moves as previously, whereas the other agent starts in a position approximately 90^o to the first agent and moves from the left-hand to the right-hand side similar to the first agent. The variation of the moving directions of the two agents is also evaluated by validating different values of the heading angle of the second agent ($\phi = 0, \phi > 0$ and $\phi < 0$, as seen in Figure 7.7).

The size of the agents is set to a bounding circle with a radius of $0.5\ m$ while the ratio $\beta_A/\alpha = 5$ and the coefficient $\gamma = 1$ are used. In both scenarios, the two agents moves at the same speed of $0.5m/s$ (with $k_\omega = 4$) so that their path intersects in the middle of their way. However, with the help of repulsive forces generated by dipole field, the two agents are able to avoid the collisions (Figure 7.8). Besides, after a small deviation from the path, due to the dipole field interaction the agents turn back directly to their original paths to continue their routes towards their goals. The distance plots show that the minimum distance of two agents are remained above the agent's diameter (marked with the green line at $1.0\ m$, in Figure 7.9), thus there are no collisions present in

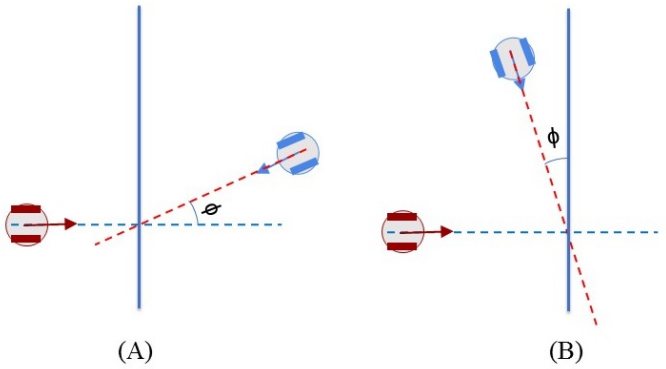


Figure 7.7: Crossing scenarios of two agents (A) Scenario 1: Two agents move toward each other with opposite directions and (B) Scenario 2: Two agents move toward each other with the heading angles of around 90^o.

the presented cases.

7.3.3 Dipole flow field for multi-agent and human-agent interaction

In the first part of this section, the behaviours of multiple agents within dipole-flow field are analysed. In the second part, the comprehensive evaluation of the dipole-flow field with the appearances of both agents and humans are preformed. Also, in the second part, the concluding experiment, which demonstrates the behaviour of the agents in presence of human in a large and realistic area, is presented.

Four agents, positioned at different orientations with the same distance to the centre of the map, take part in the first testing scenario (Figure 7.10). All agents are planned to cross the centre, and move towards their goals symmetrical to their starting positions. The agents travel within a binary map of a size of 50 x 50 m with a resolution of 10 pixels per meter and with static obstacles so that the free-space of moving and avoiding other moving objects is limited.

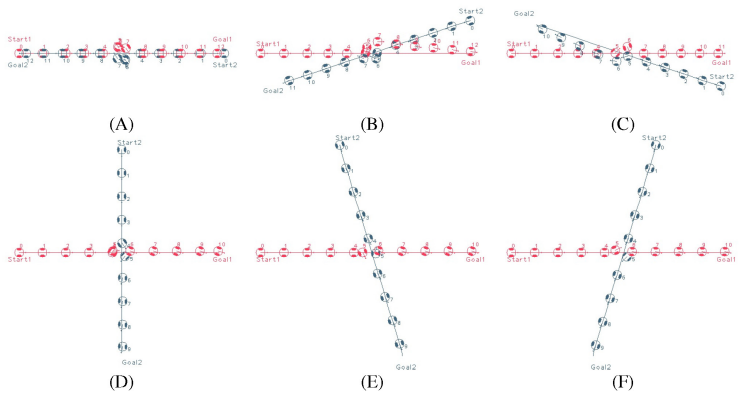


Figure 7.8: Trajectories of two agents in two scenarios. The first rows visualise the moving behaviors in dipole field of agents in Scenario 1 with different value of ϕ, (A) $\phi = 0$, (B) $\phi > 0$, and (C) $\phi < 0$. Similarly, the second rows show the results of Scenario 2 with (D) $\phi = 0$, (E) $\phi > 0$, and (F) $\phi < 0$. The time indices are used to show the location of agents at every 10 seconds.

Also, the way to reach the goal is narrowed down and there is a traffic circle in the centre of the map. Each agent has a radius of $0.5\ m$ and a moving speed of $0.5m/s$ (with $k_\omega = 4$). The quantitative measurement of obstacle avoidance (with $\beta_A/\alpha = 5$ and $d_0 = 25$) is given by measuring the minimum distance among agents over time . The closest distance of two agents when they are moving if smaller than their size will reveal a collision between them.

As depicted in Figure 7.10A, using flow-field navigation all agents are able to reach their goals. However there are existing collisions between agents (1)-(4), (2)-(4), and (3)-(4) (Figure 7.11A) with regards to the agent's radius of $0.5\ m$. With dipole-flow-field navigation, agents show ability to avoid possible collisions (Figure 7.11B). Finally, the control factor (γ) in dipole-flow field is evaluated to show its effects on the trajectories of agents in Figure 7.10C and the results in Figure 7.11A. When $\gamma < 1$ is used, the collisions are prevent in a better way by keeping the minimum distances among agents bigger. It is

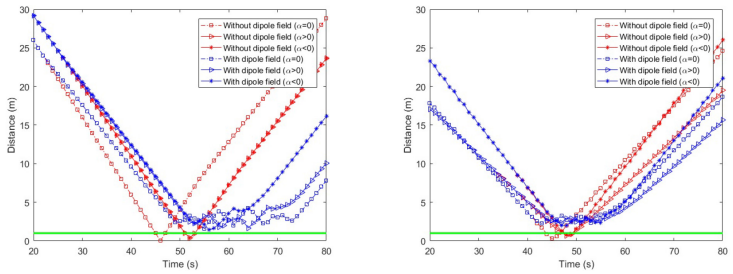
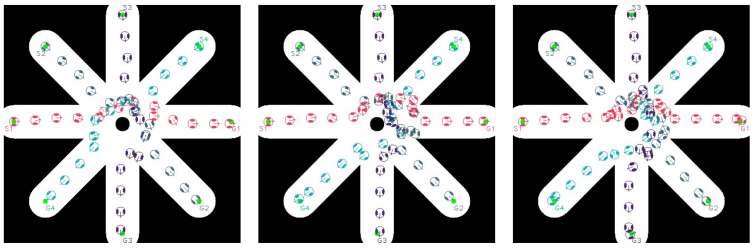


Figure 7.9: Distance of two agents over time in (A) Scenario 1 and (B) Scenario 2. The green baseline depicts the minimum distance between agents to avoid collisions.

important to note that the trajectories of moving agents are visualised to not interfering with any static obstacles from the binary map.

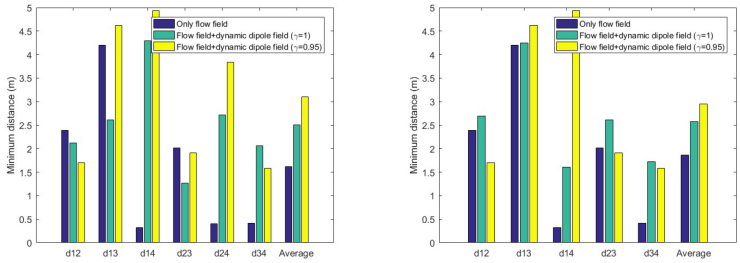
In order to evaluate dipole-flow-field for human-agent interaction, Agents 2 and 4 are replaced by two human subjects, which move as their agent counterparts, without caring the conflicts with agents. The moving trajectories of Agents 1 and 3 are described in Figure 7.12. Again, the collisions between agents are eliminated when agents are routed by the forces generated by dipole-flow field.

Finally, a general assessment of the dipole flow field for agent-agent and human-agent interactions within a large and complex binary map of variations of static obstacles drawn from a real building is presented. There are five moving agents and three human subjects in this evaluation. The map represents a part of the floor of a real building (width and length = 200 x 200 m with the same resolution of 10 pixels per meter). All agents have a bounding circle with a radius of 0.5 m, while $\beta_A/\alpha = 50$, $d_0 = 25$, and $\gamma = 0.95$ are applied with bigger values than those of the previous experiments to help agents prevent collision from a further distance. An example of moving trajectories of different agents with human is shown in Figure 7.13. The experiments were



(A) (B) (C)

Figure 7.10: Trajectories of multiple agents moving (A) without dipole field, (B) with dipole field $\gamma = 1$, and (C) with dipole field $\gamma = 0.95$.



(A) (B)

Figure 7.11: Minimum distance of agents over time in (A) multi-agent, and (B) human-agent interactions within dipole-flow field.

repeated 100 times, and for each trial start and goal positions of both agents and humans were randomised. The requirement for finding the start position and goals was that the pairwise distances among them should be at least $2.0\ m$. The speed of the agents and human during the experiment is randomly assigned within a range of $0.5 - 1.5\ m/s$ using a uniform distribution (with $k_\omega = 4$). For each trial, agent-agent and human-agent the distances are recorded for the evaluation purposes. The overall result is summarised in Table 7.2.

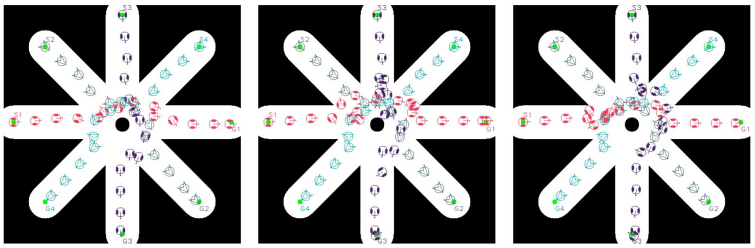


Figure 7.12: Trajectories of multiple agents moving and interacting with human (A) without dipole field, (B) with dipole field $\gamma = 1$, and (C) with dipole field $\gamma = 0.95$.

Table 7.2: Evaluation of the minimum, and average, of agent-agent and human-agent distances over all 100 trials.

Minimum of agent-agent distances (m)	Minimum of agent-human distances (m)	Average of minimum agent -agent distances (m)	Average of minimum agent -human distances (m)
2.4	1.0	10.0	8.8

7.4 Conclusion and Discussion

This paper has introduced a novel path planning algorithm for agents surrounded by static and multiple moving objects, including other robotic agents as well as humans subjects, all populating a realistic working space. The algorithm is able to process path planning in real-time by developing a navigation field so that the movements of agents is just simply controlled by the forces generated from this field. The attractive forces that drive the agents toward their desired goals are created by a static flow field. Simultaneously, the repulsive forces that prevent agent-agent, and human-agent, collisions are generated by a magnetic field of dipoles. The combination of the static flow field and dipole field forms a force to determine the moving directions of the agents at a

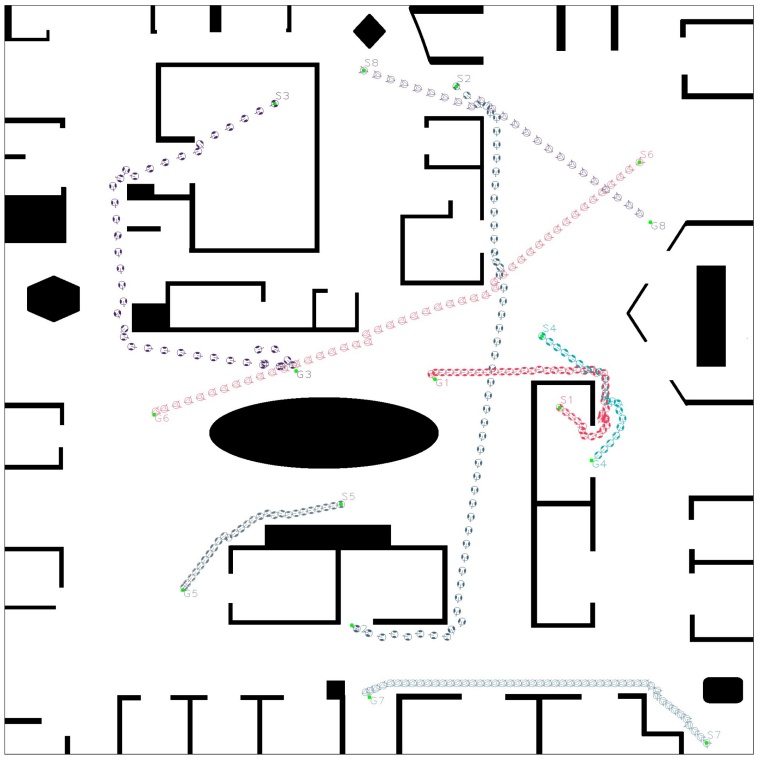


Figure 7.13: Dipole-flow field to control movements of multiple agents with the presence of three human in a 200 x 200 m large map which is visualised from a real working space. All agents are able to reach their goals with different speeds. While moving to goals, the two agents with indices 2 and 3 try to avoid the collision with human with index 6. Meanwhile, two agents with indices 1 and 4 also change directions to avoid collision with each other. In the case of the agent with index 3, the goal $G3$ is very close to the moving trajectory of human, therefore its way to the goal seems to be blocked until human with index 6 passes through $G3$. In consequence, the agent 3 must go back and later turn around to reach its goal. This behavior of moving is quite different with the scenario described in Figure 7.10B.

specific time instance.

The evaluation of the proposed approach with the static flow field, dipole field and their combinations are conducted with distinctive experiments. With static flow field, it is obvious that an agent is able to move to its goals in a binary map of static obstacles with a minimum number of re-initialising the global path using the Theta* algorithm. As can be seen in Table 7.1, the number of running Theta* instances except the first initiation is less than one time in average (if the window effects of the static flow field is set to at least twice the size of the agents, $W \geq 2S$). However, an unnecessary large window may cover otiose areas that affect the static flow field, leading to the trap of agent into a corner of the map. Therefore, the window size $W = 2S$ is recommended to increase the robustness of the static flow field and to avoid the possibility of local traps.

Within the combined dipole-flow field, the robotic agents are well routed to their destinations, while possible collisions with other agents and human are taken into account. Regarding overall evaluation of the dipole-flow field to navigate agents in a complex scenario, the average minimum distance between any two agents remains at least double the radius of bounding circle, which indicates that there are no collisions (Table 7.2). The minimum human-agent distance is $1.0\ m$. However, such a recorded observation in which the human-agent distance is close to $1.0\ m$ is only one case in 1500 obtained distance pairs (there is a group of five agents and three human subjects in the experiment so that the obtained pairs of human-agent is 1500 over 100 trials). Regarding the size of the human subjects, the bounding circle radius can be configured even less than $0.5\ m$, therefore it can be concluded that no occlusions happen in any of the simulation runs.

Recently, the aim of this work has moved towards holistic navigation solutions in real-world problems more specifically, mobile robots in densely populated areas such as, offices, and heavy vehicles in restricted spaces. Thus, by adding a control mechanism for the velocity e.g., decreasing the speed to avoid possible collisions, as well as other measures will be investigated. Be-

sides controlling the agents' velocity, the configuration of global paths with regards to multiple agents is also an important factor to ensure the reachability of all agents to their goals. In the current approach, only one optimal path to the goal is configured for each agent, without considering the conflicts with others. If any two agents enter into a very narrow area on opposite directions, as described by Kimmel and Berris [33], the dipole forces mainly push them away to avoid collisions but not help them build new paths to their goals. Therefore, the two agents tend to follow the same planned paths again and again, leading to a deadlock situation. The proposed approach could be improved by setting multiple paths for each agent. Upon evaluating the location information of others and the binary map of environment, an agent is able to decide which path, even not optimal, it should follow to reach its goal. The aforementioned problem could be also addressed by exchanging information of planned paths among agents. All agents will negotiate to optimise the flow field on a global scale to avoid any deadlock situation. However, in this case the communication protocol will become more complicated and extra processing is needed at each agent side to optimise the global path with respect to the presence of other agents. Finally, the work will be extended with different classes of agents [27], and with multiple heuristics of A* [28] to allow more thoroughly investigation of the dependability factors, and constraints on the path planning problems. The intention is also to validate the algorithm using robots and humans in outdoor settings, that resemble the qualities of construction sites.

Acknowledgments

The research leading to the presented results has been undertaken within the research profile DPAC - Dependable Platform for Autonomous Systems and Control project, funded by the Swedish Knowledge Foundation.

Bibliography

[1] H. Hu and M. Brady, "Dynamic global path planning with uncertainty for mobile robots in manufacturing," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 5, pp. 760-767, October 1997.

[2] F. Belkhouche, "Reactive path planning in a dynamic environment," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 902-911, August 2009.

[3] K. Ok, S. Ansari, B. Gallagher, W. Sica, F. Dellaert, and M. Stilman, "Path planning with uncertainty: Voronoi uncertainty fields," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Karlsruhe, Germany), pp. 4581-4586, May 06-10, 2013.

[4] Y. Golan, S. Edelman, A. Shapiro, and E. Rimon, "Online robot navigation using continously updated artificial temperature gradients," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1280-1287, 2017.

[5] Y. Wang, and G. S. Chirikjian, "A new potential field method for robot path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (San Francisco, CA), pp. 977-9822, April, 2000.

[6] A. Nash, K. Daniel, S. Koening, and A. Felner, "Theta*: Any angle path planning on grids," *Journal of Intelligent Robot System*, vol. 39, pp. 533-579, 2010.

[7] L. Valbuena, and H. G. Tanner, "Hybrid potential field based control of differential drive mobile robots," *Journal of Intelligent Robot Systems*, vol. 68, no. 3, pp. 307-322, December 2012.

[8] L. A. García-Delgado, J. R. Noriega, D. Berman-Mendoza, A. L. Leal-Cruz, A. Vera-Marquina, R. Gómez-Fuentes, A. García- Juárez, A. G. Rojas-Hernández, and I.E. Zaldívar-Huerta, "Repulsive function in potential field based control with algorithm for safer avoidance," *Journal of Intelligent Robot Systems*, vol. 80, no. 1, pp. 59-70, October 2015.

[9] J. V. D. Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Pasadena, CA, USA), pp. 1928-1935, May 19-23, 2014.

[10] E. Owen and L. Montano, "Motion planning in dynamic environments using the velocity space," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*,(Edmonton, Canada), pp. 2833–2838, August 2–6, 2005.

[11] E. Owen and L. Montano, "A robocentric motion planner for dynamic environments using the velocity space," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Beijing, China), pp. 4368–4374, October 9–15, 2006.

[12] B. Damas and J. Santos-Victor. "Avoiding moving obstacles: The forbidden velocity map," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4393-4398, October 11-15, 2009.

[13] J. V. D. Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3475–3482, May 9-13, 2011.

[14] D. Wilkie, J. V. D. Berg, and D. Manocha, "Generalized velocity obstacles," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5573-5578, October 11-15, 2009.

[15] B. H. Lee, J. D. Jein, and J. H. Oh., "Velocity obstacles based local collision avoidance for holonomic elliptic robot," *Journal of Autonomous Robots*, vol. 41, pp. 1347-1363, 2017.

[16] J.-K. Yoo and J.-H. Kim, "Navigation framework for humanoid robots integrating gaze control and modified-univector field method to avoid dynamic obstacles," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1683-1689, October 18-22, 2010.

[17] T. H. Cormen, C. E. Leisercon, R. L. Rivest, and C. Stein, "Introduction to algorithms," Third Edition, The MIT Press, 2009.

[18] D. S. Yershov and S. M. LaValle, "Simplicial Dijkstra and A* Algorithms for optimal feedback planning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3862-3867, September 25-30, 2011.

[19] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A*," *Journal of Artificial Intelligence*, vol. 155, no. 1-2, pages 93-146, 2004.

[20] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354-363, 2005.

[21] D. Ferguson and A. Stentz, "Using interpolation to improve path planning: The field D* algorithm," *Journal of Field Robotics*, vol. 23, no. 2, pp. 79-101, 2006.

[22] X. Sun, W. Yeoh, and S. Koenig, "Dynamic fringe-saving A*," *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, pp. 891-898, May 10–15, 2009.

[23] S. Koenig, M. Likhachev, Y. Liu, and D. Furcy, "Incremental heuristic search in AI," *Journal of AI Magazine*, vol. 25, no. 2, pp. 99-112, 2004.

[24] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3310-3317, May 8–13, 1994.

[25] P. Yap, N. Burch, R. Holte, and J. Schaeffer, "Block A*: Database-driven search with applications in any-angle path-planning," in *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pp. 120-125, May 7-11, 2011.

[26] T. Uras and S. Koenig, "An empirical comparison of any-angle path-planning algorithms," in *In Proceedings of the Symposium on Combinatorial Search (SOCS)*, pp. 206-210, 2015.

[27] D. Panagou, "A distributed feedback motion planning protocol for multiple unicycle agents of different classes," *IEEE Transactions on Automatic Control*, vol. 3, no. 62, pp. 1178-1193, 2017.

[28] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic A*," *International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 224-243, 2016.

[29] J. J. Rubio, E. Garcia, G. Aquino, C. Aguilar Ibañez, and J. Pacheco and A. Zacarias, "Learning of operator hand movements via least angle regression to be teached in a manipulator", *Evolving Systems*, pp. 1-16, 2018.

[30] J. J. Rubio, E. Garcia, and J. Pacheco, "Trajectory planning and collisions detector for robotic arms," *Neural Computing and Applications*, vol. 21, no. 8, pp. 2105-2114, 2012.

[31] S. R. Munasinghe, C. Oh, J.-J. Lee, and O. Khatib, "Obstacle avoidance using velocity dipole field method," in *Proceedings of the International Conference on Control, Automation, and Systems*, pp. 1657-1661, 2005.

[32] T. Igarashi, Y. Kamiyama and M. Inami, "A dipole field for object delivery by pushing on a flat surface," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5114-5119, May 2010.

[33] A. Kimmel and K. Berris, "Decentralized multi-agent path selection using minimal information," in *Proceedings of the 12th International Symposium on Distributed Autonomous Robotic Systems*, pp. 341-356, January 2016.

[34] P. Morin and C. Samson, "Chapter 43: Motion control of wheeled mobile robots," Springer Handbook of Robotics, Springer, 2008.

Chapter 8

Paper B: Dependable Navigation for Multiple Autonomous Robots with Petri Nets Based Congestion Control and Dynamic Obstacle Avoidance

Lan Anh Trinh, Mikael Ekström, and Baran Cürüklü
Submitted to Journal of Intelligent and Robotic Systems, Springer, 2021.
(Minor revision)

Abstract

In this paper, a novel path planning algorithm for multiple robots using congestion analysis and control is presented. The algorithm ensures a safe path planning solution by avoiding collisions among robots as well as among robots and humans. For each robot, alternative paths to the goal are realised. By analysing the travelling time of robots on different paths using Petri Nets, the optimal configuration of paths are selected, the prime objective is to avoid congestion when routing many robots into a narrow area. The movements of robots are controlled at every intersection by organising a one-by-one passing of the robots. Controls are available for the robots which are able to communicate and share information with each other. To avoid collision with humans and other moving objects (i.e. robots), a dipole field integrated with a dynamic window approach is developed. By considering the velocity and direction of the dynamic obstacles as a source of a virtual magnetic dipole moment, the dipole-dipole interaction between different moving objects will generate repulsive forces proportional to the velocity to prevent collisions. The whole system is presented on the widely used Robot Operating System (ROS) platform so that its implementation is extendable to real robots. Analysis and experiments are demonstrated with extensive simulations to evaluate the effectiveness of the proposed approach.

8.1 Introduction

Path planning is one of the key components of a robotic control system, with the function to drive a robot from a starting to a goal position without crossing through obstacles, and do this in an optimal manner. Although the path planning problem has been addressed since the 1950s [11, 12], it has not been solved thoroughly until today [13]. Over the years many solutions have been restricted to static environments or single robot cases. In general, the path is optimised with respect to a cost function, which primarily is based on the total length of the path, and in some cases other factors, such as the complexity of the terrain, areas of interest, and other environmental factors. Recently, new challenges have emerged for path planning of fully autonomous robots within the Industry 4.0 context [39] where a mobile industrial robot is no longer separated from humans in space. More specifically, the robot shares the workshop floor with workers, and it interacts with them as well as other robots. The navigation is crucial in such problem domains, since these scenarios demand collision avoidance capability even if there are only means to predict the exact movement of moving obstacles to a certain degree. Thus, when a path planning algorithm is designed, industrial standards should be considered. Dependability is important in industrial system engineering with its focus on a system's (i) availability, (ii) reliability, (iii) safety, and other properties such as (iv) maintainability, (v) integrity, etc.

In this paper, a dependable path planning algorithm is evaluated with respect to the first of those properties, i.e the points (i)-(iii). *Reliability* is assessed by the correctness of the control system to drive a robot to the goal. *Availability* is reflected by continuity of the system within a time limit, not having the dead-lock situation which happens when the robot takes a very long time to reach a target, or even not able to do so. Lastly, *safety* is evaluated by the obstacle avoidance abilities of the robot to prevent collisions with static and moving objects.

In general, a path planning algorithm is separated into global and local

path planning. The former with the target to find a way from a start position to a goal having no intersection with any static obstacles in the map, while the latter will help the robot to avoid collisions with moving obstacles. So far, numerous global and local path planning algorithm have been developed with the ambition to improve dependability of navigation systems. Efforts have been made with the global path planning to find a feasible and effective route for a robot [34, 35, 36]. In many cases, a map of configuration space is represented as a graph and a graph-based searching algorithms like A*, is used to find a collision-free path [12, 37]. The Theta* algorithm [22] adds line-of-sight checks into A* so that the path to the goal becomes more straight, thus contains fewer turns. Dynamic A* adapts to the changes of a static map in real-time, allowing the path planning to be more robust in unknown environments. Other common approaches explore the map by sampling, such as probabilistic road map (PRM) [28], with the goal of building the map by one-time sampling or rapidly-exploring random trees (RRTs) [27] to generate samples incrementally from the initial position until reaching the goal.

For multiple robots, the multi-agent path finding (MAPF) algorithms have been developed to search for optimal paths to drive the robots to their goals without conflicts. The conflicts happen when several robots occupy the same place at an instance of time. In general the MAPF problem is nondeterministic polynomial (NP) hard [7] where the searching space grows exponentially with respect to the number of robots. A*-based search [1, 2] was one of early solutions to perform a search with heuristics in higher dimensional space with multi robots. Conflict-based search (CBS) [6] can be used to find an optimal solution by a repeated process. First, CBS initialise a root node including all paths as the shortest paths of robots to the goal without concerning other robots. Continuously CBS expands the tree search starting from the root node by adding child nodes with conflicting constraints to avoid collisions. Finally, the constraints about potential conflicts are used to refine the paths at each node. Enhanced CBS [3] solved the MAPF problem within a suboptimal bound. Cooperative A* (CA*) [5], also a suboptimal and incomplete approach, prioritises the robot

to assign the path by checking the number of collisions of the shortest path as compared to the planned paths. Priority-based search (PBS) [4] is the combination of CBS and CA* to improve the priority order of CA*. In overall, there have been a lack of a MAPF solver dealing with complex constraints like allowing several robots to meet at an intersection or one robot to wait for others to pass through the intersection. This paper investigates an effective tool to analyse and find optimal settings for global paths of multiple robots on those situations.

Once the global path to the goal has been generated, obstacle avoidance with local path planning ensures the safety for both robots and humans by preventing collisions between them. In velocity-obstacle (VO) approaches [23], a velocity obstacle is defined as a forbidden zone in the velocity space of the robot where a velocity vector in that zone could result in a collision with moving obstacles. The reciprocal velocity obstacle (RVO) [29, 30] implements VO in a distributed manner where each robot has responsibility to avoid collisions with the rest by choosing a velocity outside VOs. However, after escaping from the collision, the robot tends to come back to its preferred velocity, possibly leading into a reciprocal dance. To address this problem, the optimal reciprocal collision-avoidance (ORCA) [32] casts the solution with a linear programming optimisation. Simultaneously, the hybrid reciprocal velocity obstacle (HRVO) [31] seeks for the velocity of the robot more towards one side of the half plane of VOs. Nonetheless, VO-based methods work with the assumption that the velocity of a robot is not changed over the time interval. Therefore, these kind of approaches are more suitable to prevent potential collisions within a local range. Alternatively, a learning-based approach with reinforcement learning, has been investigated using Markov decision process to learn from history to predict the next movement of the robot with respect to observation [33]. Yet, a long training process needs to be done to collect sufficient amount of data to perform planning in both local and global scale. Conventional, however still widely applicable for local path planing, are potential field and dynamic window methods [38]. Potential field is defined as the combination of a repulsive

field to push robots away from obstacles and attractive fields to drive a robot to a goal. Meanwhile, dynamic window approach (DWA) [9] generates a set of possible future trajectories of a robot with respect to the current dynamic configuration and evaluates these trajectories to find the optimal solution. As aforementioned, the manner global and local path planning systems allowing navigation of several robots into one narrow area might make the robots to block each other, i.e, lead to dead- or live- lock situations. To address the problem of congestion when routing multiple robots into a crowded place, a group of the robots should proactively define effective routes to reduce conflicts with each other. The problem presented here has some similar key features with traffic control in urban areas. The most common method used to ensure safe transportation is to have traffic signals to control the vehicles and pedestrians through cross ways. A modern monitoring system nowadays can collect information about traffic jams and send notification to drivers to avoid moving toward a crowded area. These approaches lead to the introduction of a congestion control for path planning that is presented in this work to achieve dependability in a navigation system for autonomous robots in a dynamic environment.

Previously, with respect to solving conflicts and sharing of resources, it has been shown that Petri net (PN) provides an effective solution to such problems for an autonomous system. The uses of PN to design autonomous robots are demonstrated by Yasuda et al. [16], Iocchi et al. [17]. This is further exploered with the use of other extension with ROS on real robots by Fabre et al. [20] and also with fault tolerance by Miyagi et al. [18], Lusier et al. [19]. Although there have been previous works of using PN on path planning [10], they have mainly focused on the discrete system of robots changing states from one node to another in a static graph. Such a static configuration is highly dependent on the fixed and precise travelling time of robots among the nodes. However, this condition is hard to be satisfied for an autonomous robot at planning time when the robot must be able to recognise and work with humans and/or other robots, in an unstructured and unknown environment.

With regard to PN as an effective tool in dependable autonomous control

to resolve conflicting problems, this paper presents a new approach of using it in generating the paths of multiple robots to reduce congestion by avoiding routing many robots into the same place. There are three main contributions of the paper. First, the paper shows how to model the path finding problem for multiple robots with PN and how to analyse alternative paths of each robot to find the optimal configurations with less traffic conflicts along the paths. The optimisation on PN allows to solve a MAPF problem with complex constraint such as stopping robots at intersection to wait for other robots. Second, the paper proposes the way of monitoring moving tokens in the PN to control the traffic of robots at every path crossing to reduce conflicts. Particularly, the first contribution helps to find the optimal global path for robots, whilst the second one realises the PN control to synchronise the moving of the robots on assigned paths. Since the PN is mainly designed to control robot movements, the collisions between them and humans are addressed by DWA. However, the use of DWA is not effective if it only prevents collisions in a passive manner, meaning that a robot doesn't react until the moving obstacles have moved close enough and then considers them as normal static obstacles. This type of behaviour needs to be avoided. In the third contribution, this paper shows the implementation of the dipole field algorithm on top of the DWA algorithm to present a new method for obstacle avoidance, which is based on the information about directions and speeds of the moving objects. Moving obstacles are described by magnetic dipoles with their moments aligned with the direction of velocities. As a consequence, opposite dipoles will generate repulsive forces to turn the robot so that they do not collide with obstacles. The capability of PN control with dipole field on DWA allows the whole proposed path planning system to handle both static and dynamic obstacles for the robots' paths. In overall, the whole system is integrated in the navigation stacks of ROS, one of the most widely used robotic frameworks today.

The rest of the paper is organised as follows. Through Section 2, the problem of this work is stated, following by the proposed path planning algorithm. Thereafter, in Section 3 the simulation results to evaluate the proposed sys-

tem are presented. Finally, contributions of the paper and future directions are summarised and concluded in Section 4.

8.2 Methodology

8.2.1 Problem statement

The basic annotations and preliminaries for the above equations are defined as follows.

Robots. This work copes with a team of N identical robotic agents $\mathcal{A} = \{a_1, a_2, ..., a_N\}$ which are simply presented by a unit-circle model with the common radius r. The robot are freely moving on 2-D plane $\mathcal{M} = \mathbb{R}^2$. The location of each robot i at time t is denoted by $x_i(t) \in \mathbb{R}^2$, with corresponding velocity $\dot{x}_i(t)$. The area occupied by a robot is denoted by $\mathcal{C}_i(x_i(t)) \subset \mathbb{R}^2$, a circle at position $x_i(t)$ and radius r.

Map and static obstacles. The working environment is presented with a simple binary grid map where a set of static obstacles (white pixels) are described by $\mathcal{O} \subset \mathbb{R}^2$ and robots are freely moving within the black regions of the map $\mathcal{M} \setminus \mathcal{O}$. Let $\mathcal{O}_r$ be the subset $\mathcal{O}$ dilated by the robot's radius r. It is obvious that if there is no intersection of the moving trajectory of the robot trajectory with $\mathcal{O}_r$, then the motion of robots are collision free with respect to the static obstacles.

Humans and moving obstacles. There are M human subjects present in the working environments together with the robots. The information about current locations, planned paths, velocity, accelerations, etc. are shared among the robots, however, they are unable to know the exact trajectories of humans. The information about dynamic information of the human subjects, e.g. location $o_i^j(t)$ and velocity $\dot{o}_i^j(t)$ are only estimations that are available with every observation, where j is the index of a human observed by the robot i, $1 \leq j \leq n_i, n_i \leq M$. Assume that the human is occupying a circle $\mathcal{H}_i^j(o_i^j(t))$ with the centre at $o_i^j(t)$ and the maximum radius r_h, each robot i will then see

a set of moving obstacles/humans as $\{\mathcal{H}_i^1, \mathcal{H}_i^2, ..., \mathcal{H}_i^{n_i}\}$.

Global paths. Each robot is assigned a task to move from a starting point to a goal. Using a global guidance system, a number of alternative paths are planned. These global paths are updated frequently to adapt to the changes in robot trajectories. Let a set of global paths for a robot i be $\{\mathcal{S}_i^1, \mathcal{S}_i^2, ..., \mathcal{S}_i^k\}$ where the path consists of connected line segments. Different paths are leading into the same goal will simply extend the area $\mathcal{G}_i$, possibly defined by a circle with a predefined radius.

The problem to be addressed in the proposed work has the aim that all robots should be able to complete their navigation tasks without colliding with any static and moving obstacles in the working environment. For all times from t_i^0 to t_i^1 for a robot i to travel along its trajectory, the collision constraints are expressed by,

$$\mathcal{C}_i(x_i(t)) \cap \Bigg(\mathcal{O} \cup \Big(\bigcup_{j \in \{1,2,...,N\}, j \neq i} \mathcal{C}_j(x_j(t)) \Big) \cup \Big(\bigcup_{k \in \{1,2,...,n_i\}} \mathcal{H}_i^k(o_i^k(t)) \Big) \Bigg) = \emptyset \qquad \forall t \in [t_i^0, t_i^1] \quad (8.1)$$

which is correspondent to no intersection of the trajectory with static obstacles, other robots and moving obstacles in its environment. The requirement to reach the goal is given by:

$$\mathcal{C}_i(x_i(t_i^1)) \cap \mathcal{G}_i \neq \emptyset. \quad (8.2)$$

Routing several robots into narrow areas could lead to congestion, that may lead to a deadlock situation or prolong the time to complete the navigation task since robots need to go around to avoid collisions with each other. The main problem to be solved in this work is to tackle the issues defined by equations (8.1) and (8.2) with the use of PN to synchronise the movements of robots to find less not only collision- but also congestion-free paths to reach their goals correctly.

The solution starts with the construction of multiple paths for each robot (Section 8.2.2). A path, which is presented by a set of line segments connecting from a starting point to a goal, is found by the Theta* algorithm. Continuously, each robot broadcasts its paths to all robots within the working space. By checking the intersections of the planned paths, PNs are constructed to analyse the movements of the robots when they enter the intersection regions (Section 8.2.3). In each PN, where a single robot is correspondent to one token of the network. An optimisation algorithm is performed to choose the path for each robot to minimise the conflict/congestion and the travelling time to complete the task (Section 8.2.4). Once the optimal path is found, the robot then follows the path and synchronises with other robots to avoid collisions (Section 8.2.6). Collisions with other obstacles that are not able to communicate their planned trajectories with the robots, e.g. humans, are addressed by using DWA combined with a dipole field (Section 8.2.7). In the case that one of the robots is getting stuck, the global path planning is reactivated to generate a new path from current position to the goal, and PN planning is updated with a new moving control plan. The overall architecture of the system is described in Figure 8.1.

8.2.2 Multiple global paths

A global path planning algorithm needs information regarding the environment e.g. as a scanned map beforehand to generate a path from a starting point to a desired goal. The global path planning mainly applies for a static environment. This means that in order to compute the global path to a goal for the robot, the presence of other robots and humans, i.e. dynamic objects, are not taken into account. Theta* [22] has realised an any-angle path algorithm, which is able to find an optimal path in any direction, by adding a line-of-sight detection function to each search iteration. Unlike A* or Dijkstra, the path found by Theta* is a connection of line-of-sight nodes so that the generated path is smoother, more realistic, and has fewer turns. With regard to those advantages of Theta*, it has been used as the main global path planning in this paper. To generate

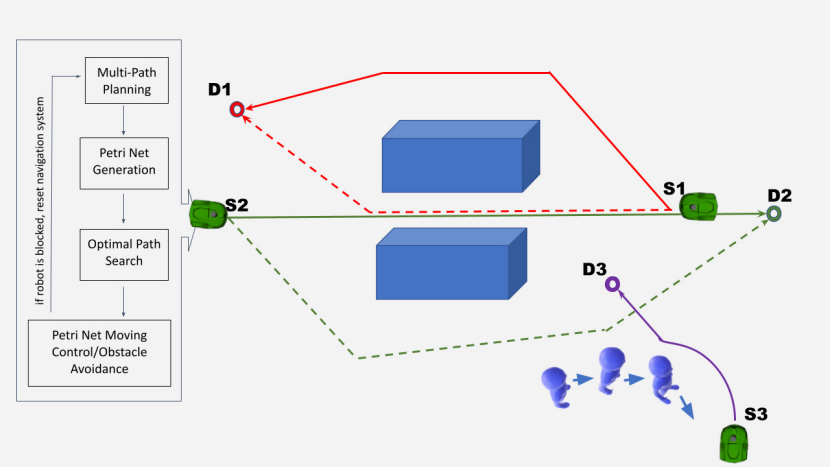


Figure 8.1: The overall architecture of the proposed navigation system.

multiple paths, Theta* is utilised to sequentially find a new path after inserting a set of found paths into an obstacle map. This way, the next found path does not overlap with the previous ones (Figure 8.2). A backup path is critical if the robot needs to pass through a narrow area with the possibility of facing another robot so that it has to find an alternative way. On an empty space, the multiple paths created by this approach can be close to each other. Therefore, to reduce the complexity of the control PN, multiple paths are only added if separation among them are big enough. The algorithm to find multiple paths from a starting point to a destination is summarised in Algorithm 2.

8.2.3 Petri net construction

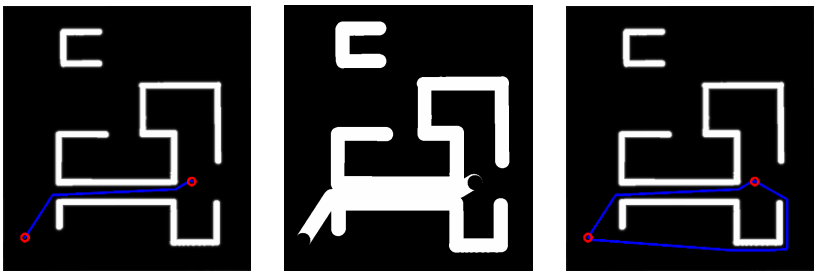
The basic definitions of a PN are given as follows.

Definition 1*(Petri net)*: PN is considered as a network of places, arcs and transitions. Mathematically, it is defined as a bipartite graph of a set of tuples $\langle P, T, W \rangle$, where $P = \{p_1, p_2, ..., p_{|P|}\}$ and $T = \{t_1, t_2, ..., t_{|T|}\}$ are disjoint sets of places and transitions, $|P|$ and $|T|$ are the number of elements of P and

Algorithm 2 Finding multiple global paths.

Input:A pair of a source s and destination d, static obstacle map $\mathcal{O}_r$, N_p maximum number of paths
Output:A set of global paths $\Xi = \{\xi_1, \xi_2, ..., \xi_n\}$ from s to d, $n \leq N_p$

$\Xi \leftarrow \emptyset$
$i \leftarrow 0$
$is\_find\_next \leftarrow True$
while $i \leq N_p \wedge is\_find\_next$ **do**
  Find a new global path ξ_i from s to d on $\mathcal{O}_r$ using Theta*
  for $\xi_j \in \Xi$ **do**
    Present global paths by a sequence of points in 2-D (dimensional) space, $\xi_i = \{s_1^i, s_2^i, ..., s_{l_i}^i\}$, $\xi_j = \{s_1^j, s_2^j, ..., s_{l_j}^j\}$ where l_i, l_j are the number of points of the two paths respectively
    $d_1^i, d_2^i, ..., d_{l_1}^i \leftarrow$ a set of minimum Euclidean distance of each point s_i^i to ξ_j
    $d_1^j, d_2^j, ..., d_{l_j}^j \leftarrow$ a set of minimum Euclidean distance of each point s_j^j to ξ_i
    $d(\xi_i, \xi_j) = \max\{d_1^i, d_2^i, ..., d_{l_i}^i, d_1^j, d_2^j, ..., d_{l_j}^j\}$
    if $d(\xi_i, \xi_j) < threshold$ **then**
      $is\_find\_next = False$
      break
    else
      Dilate ξ_i by the radius of the robot and add it into $\mathcal{O}_r$ $\Xi := \Xi \cup \xi_i$,
    end if
  end for
end while



(A) (B) (C)

Figure 8.2: Multiple path generation. (A) The first path created by Theta*. (B) Inserting the first path into the binary obstacle map. (C) The second path also made by Theta* but separated from the first one.

T respectively, and $W \subseteq (P \times T) \cup (T \times P)$ is a set of arcs connecting from a place to transition and vice versa. □

In a PN, an input arc of a transition is defined as an arc that runs out from a place while the contrasting one, i.e, the output arc goes out from a transition to a place. Both the input and output flows of the transition are added by positive weights. With regard to a set of output weights O and a set of input weights I, PN is described as a set of five tuples $\mathcal{G} = \langle P, T, W, O, I \rangle$. Places in a PN may consist of a number of marks named tokens.

Definition 2 *(Marking)*: The marking M is expressed as a vector $[M(p_1), M(p_2)$ $, ..., M(p_i), ..., M(p_{|P|})]^T$, in which p_i is a place, $|P|$ is the number of places in PN, and $M(p_i)$ is the number of tokens at the place p_i. □

Let $\mathbf{O}$ be a two dimensional matrix of weights $O(p_i, t_j)$ from the place p_i to the transition t_j. $\mathbf{I}$ is similarly defined by the weight $I(t_j, p_i)$ from the transition t_j to the place p_i. It is noted that $1 \leq j \leq |T|$, where $|T|$ is the total number of transition. Thus, a change of the marking vector of a transition from

M to M' is given by a finite sequence of transitions

$$M'(p) = M(p) + I(t,p) - O(p,t), \forall p. \tag{8.3}$$

It is said that the marking M' is reached by the marking M by firing t. In the vector form, this transition marking is presented as

$$M' = M + I_p - O_p, \tag{8.4}$$

where $I_p = \mathbf{I}(\cdot, p)$ is the column vector p-th of the matrix $\mathbf{I}$, and $O_p = \mathbf{O}^T(p, \cdot)$ is of $\mathbf{O}$. In general, if the marking M' is reachable from M by a finite sequence of k transitions $\sigma = t_{i_1} t_{i_2} ... t_{i_k}$, then (8.4) is rewritten as

$$M' = M + \mathbf{C} \cdot \varsigma \tag{8.5}$$

where $\varsigma = [\sigma_1, \sigma_2, ..., \sigma_{|T|}]^T$ is a vector where each element σ_j counts the number of times t_{i_j} appears in the sequence σ. An example of a PN described in Figure 8.3 shows the changes of the marking from $M = [1,2,0]^T$ into $M' = [0,1,2]^T$ after the transition t_0 fires.

(A) (B)

Figure 8.3: A PN example with three places p_0, p_1 and p_2 and one transition t_0. (A) A PN with an enabled transition. (B) The PN after the transition fires.

With an initial marking M_0, the full description of a PN consists of six tuples $\mathcal{G} = \langle P, T, W, I, O, M_0 \rangle$. A full graph of all possible markings and

transitions, i.e. a reachability set, is described by state-space analysis. One of the methods to construct a PN for the path planning problem is to divide the working space into non-overlapping regions with a grid layout where each cell of the grid corresponds to a place of the PN [15]. However, a robot usually crosses many grid cells along its trajectories and some of them never play a role in controlling congestion as they are not the junctions of two or more robots' trajectories. This leads to unnecessary defined spaces and transitions, resulting in a very sparse PN to control robot movements. In this paper, the place is created only at every intersection of the global moving paths of different robots. Each place of the PN corresponds to a region of intersection and is assigned the location at the centre of that region. In order to find the area of intersection, the thickness of the path is dilated by the radius of the robot. For a pair of paths, there are one or more intersections formed along the paths. Due to the dilation of the path, an intersection is not presented as a point, but a polygon. Consequently, small polygons are filtered out based on their perimeters. The centre of a polygon is estimated by averaging all middle point of the segments passing through the polygon. PN is applied to avoid the congestion by granting permission to just one robot to pass through an intersection at an instance of time. The trajectory crosses that happen inside a narrow area may even lead to an unexpected situation of making robots getting locked and not able to escape from the congestion area. Therefore, other control places are created to handle such an area to limit the number of robots allowed to pass through it. For simplicity, those types of areas are manually given in this work by providing a list of polygons to cover the restricted areas. Meanwhile, it is also inefficient to have control places which are very close to each other. To address this issue, based on the positions of the intersection, the places are grouped by hierarchical agglomerative clustering [8] until the distances among every places are greater than a predefined threshold. It is also important to avoid creating places nearby the sources, goals, and the restricted areas. The combination of merging the paths and remaining only the important control places helps to create a compact PN to control the movements of robots. In summary, a set

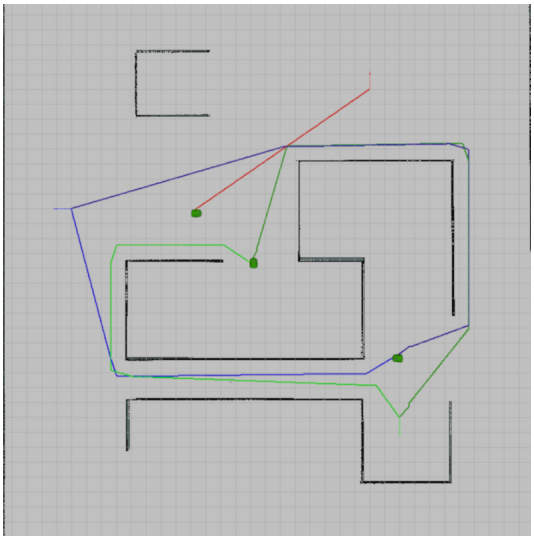
of intersections are estimated by Algorithm 3. An example of creating control place is described in Figure 8.4 with three control places numbered 6, 7, and 8, respectively. The three green circles present the starting positions while the three blue circles are the destinations. Three control places are marked with the red circles. Control place 6 lies at the middle of the intersected area of two paths. Meanwhile, the control place 7 is formed by grouping the intersection from three paths. The restricted area 6 is defined to be inside a corridor to allow only one robot to pass through at a time. Although there is an intersection close to the restricted area 6, no control place is created as it stays on the entrance (or the exit) of the corridor.

Algorithm 3 Finding intersections of global paths.

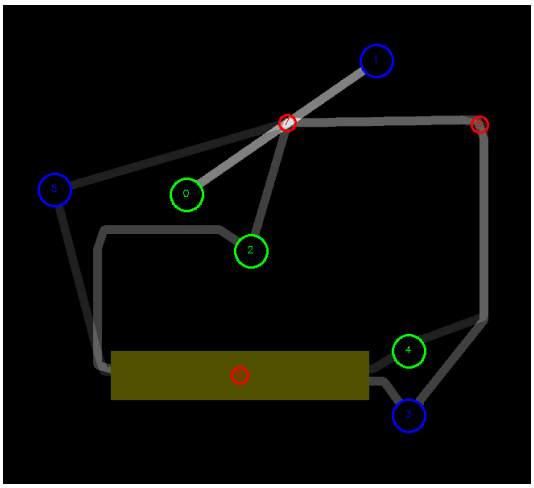
Input: A set of global paths $\Xi = \{\xi_1, \xi_2, ..., \xi_n\}$ planned for every robots of the team and a set of restricted areas $\mathcal{R} = \{R_1, R_2, ..., R_r\}$
Output: All intersections $\Upsilon = \{I_1, I_2, ...\}$ among global planned paths

Let dilate each path $\xi_i, \forall i \in [1, n]$ by the radius of the robot
$\Upsilon \leftarrow \emptyset$
for $\xi_i \in \Xi$ **do**
  for $\xi_j \in \Xi$ **do**
    Let I_{ij} be the intersected polygon of two paths ξ_i and ξ_j
    if $perimeter(I_{ij}) > \delta_p \wedge distance(I_{ij}, source/goal) > \delta_{sd} \wedge distance(I_{ij}, R_k) > \delta_r (\forall R_k \in \mathcal{R})$ **then**
      $\Upsilon := \Upsilon \cup \{I_{ij}\}$
    end if
  end for
end for
$HierarchicalAgglomerativeClustering(\Upsilon)$
$\Upsilon := \Upsilon \cup \mathcal{R}$

Considering a cross as a space resource allocated to a single robot each time, the PN model is constructed by Algorithm 4 to synchronise the movements of multiple robots. In this algorithm, a place of PN is added according to a cross and a transition represents a physical connection between two adjacent crosses on a path. An example to add a PN place to control the movements



(A)



(B)

Figure 8.4: The creation of control places from the intersection of global paths. (A) Two robots have two alternative paths to their goals (the green and blue paths) while the other has only one path (the red one). (B) The corresponding control places.

of two robots through a cross is described in Figure 8.5. The places assigned for the moving paths of Robot 1 and Robot 2 are named by $P1\_\{index\}$ and $P2\_\{index\}$ respectively (the places $P1\_2$ and $P1\_3$ for Robot 1, $P2\_2$ and $P3\_3$ for Robot 2). The control place C is made to manage two robots to pass through the intersection. Regarding the location of those PN places on the real global paths, for every intersection $\forall j, I_i^j$, the distances from p_i^{2j} and p_i^{2j+1} to I_i^j are set to be equal to D_i^{2j} and D_i^{2j+1}, where those distances are estimated by the Euclidean distance from p_i^{2j}, p_i^{2j+1} to the centre of I_i^j. Therefore, p_i^{2j} and p_i^{2j+1} lies on the circles which have the centre at the middle of I_i^j and radius D_i^{2j}, D_i^{2j+1} (Figure 8.6.A). For restricted areas (Figure 8.6.B), p_i^{2j} and p_i^{2j+1} are on the circles which have the same radius D_i^{2j}, D_i^{2j+1} but their centres stay at the crossing points between the path ξ_i and I_i^j. Some special cases are described in Figure 8.7. In the first situation (Figure 8.7.A) where the separation of two intersections is less than the total distance $D_i^{2j} + D_i^{2j+1}$, p_i^{2j} and p_i^{2j+1} are placed on the same location with the distance D_i^{2j} away from the second intersection. In another case described by Figure 8.7.B, p_i^{2j} and p_i^{2j+1} are placed at the starting or the ending point of the path.

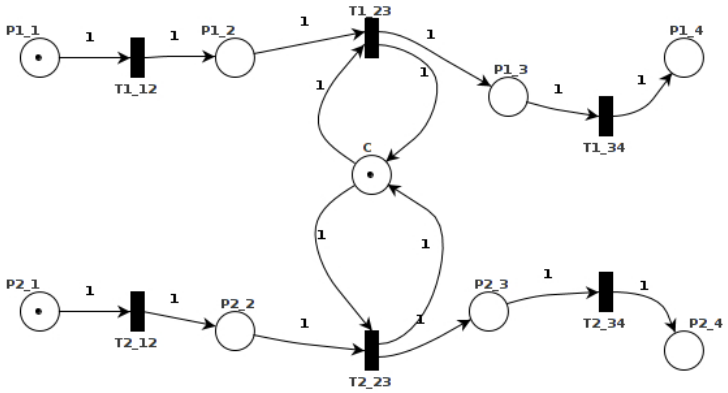


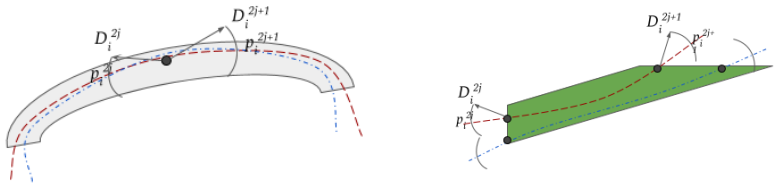
Figure 8.5: An example of creating a control place to synchronise the movements of two robots.

Algorithm 4 Construction of PN model.

Input:A team of robot with initial positions, a set of global paths $\Xi = \{\xi_1, \xi_2, ..., \xi_n\}$ planned for every robots of the team and the intersection set Υ

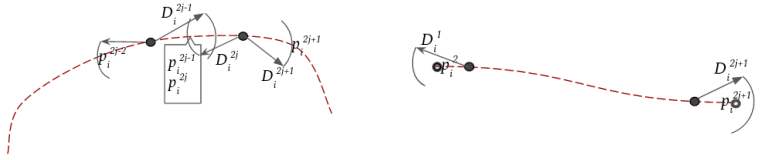
Output: The PN model (P, T, W, M_0) of the team

Let $P \leftarrow \emptyset$, $T \leftarrow \emptyset$, $W \leftarrow \emptyset$, $M_0 \leftarrow 0$
for $\xi_i \in \Xi$ **do**
  Find a set of intersections $I_i^1, I_i^2, ..., I_i^j, ..., I_i^{n_i} \in \Upsilon$ which are crossed by ξ_i
  Add two places p_i^{2j} and p_i^{2j+1}, before and after an intersection I_i^j, into PN $P = P \cup \{p_i^2, p_i^3, ..., p_i^{2j}, p_i^{2j+1}, ..., p_i^{2n_i+1}\}$ for every $\{I_i^j | \forall j\}$
  Add source place p_i^1 and destination place $p_i^{2n_i+2}$
  Add one token into the starting place $M_0[p_i^1] = 1$
  for $j \leftarrow 1$ to $2n_i$+1 **do**
    Add transition $t_{i,j}$ to T
    $\tau_{i,j}$, the delayed time at $t_{i,j}$, is approximated by the travelling distance between two places
    $W := W \cup \{(p_j, t_{i,j}), (t_{i,j}, p_{j+1})\}$
  end for
end for
Find all cells $\mathcal{C} = \{c^1, c^2, ..., c^{n_C}\}$ that are passed by at least two paths
for $c^i \in \mathcal{C}$ **do**
  Add a control place into PN $P = P \cup \{c^i\}$
  Add one token into this control place $M_0[c^i] = 1$
  Add transitions to connect c^i with related places with a description given in Fig. 8.5
end for



(A) (B)

Figure 8.6: The position of p_i^{2j} and p_i^{2j+1} before and after the control place. (A) For intersection outside restricted areas. (B) For restricted areas.



(A) (B)

Figure 8.7: The position of p_i^{2j} and p_i^{2j+1} under special cases. (A) The distance between two intersections is less than $D_i^{2j} + D_i^{2j+1}$. (B) The intersection with the distance to a start of a path is less than D_i^1 or the distance to the end of the path is less than D_i^{2j+1}.

8.2.4 Estimation of optimal configuration

When a path has many crossings, it means that the robot's travelling time is prolonged. Therefore, the PN construction is improved by first setting multiple paths for each robot to its goal and then choosing the optimal configuration of PN to minimise the travelling time of robots and to avoid deadlock situation. In Algorithm 4, $\mathcal{T} = [\tau_1, \tau_2, ..., \tau_{|T|}]^T$ is the approximation of the delayed travelling time over a transition. Assume that all robots are configured to have the same speed, the travelling time is therefore proportional to, or for simplicity is equal to, the travelling distance from one transition into another. With regards to $\mathcal{T}$, the firing rule is modified to consider the delayed time in the transition firing: After a transition t_i is enabled, it will fire after τ_i time units. Those extensions actually introduce timed PN with an associated finite firing duration to each transition in PN. Assume that after n intermediate markings, all the moving tasks are accomplished from the initial marking M_0. Let $M_j, j = 1, 2., .., n$ denote a sequence of markings and the firing count vector ς_j that makes the moves of one marking to the next, $M_j = M_{j-1} + \mathbf{C} \cdot \varsigma_j, j = 1, 2, ..., n$. The total of the corresponding travelling times of all robots after firing the sequence $\varsigma_j, j = 1, 2, ..., n$ is estimated by $\sum_{j=1}^{n} f(\varsigma_j)$, where $f(\varsigma_j)$ is the delayed time of the whole PN after ς_j is performed. For simplicity, each robot is setup with the same number, k, of alternative paths to reach the goal. Let $z_{i,p}$ be a set of $N \times k$ binary variables defined according to how each robot choose its path,

$$z_{i,p} = \begin{cases} 1 & \text{if the robot } i \text{ selects the path } p \\ 0 & \text{otherwise}, \forall p = 1, 2, ..., N. \end{cases} \tag{8.6}$$

The constraint that only one path is assigned to a robot is expressed by,

$$\sum_{p=1}^{k} z_{i,p} = 1. \tag{8.7}$$

Therefore, $\varsigma_j(z_{i,p})$ is a firing sequence decided by the selection of $z_{i,p}$. The optimal problem to find best assignments of paths to robots is stated as follows,

$$\begin{aligned} & \underset{z_{i,p}}{\operatorname{argmin}} && \sum_{j=1}^{n} f(\varsigma_j) \\ \text{s.t.} \quad & && M_j = M_{j-1} + \mathbf{C} \cdot \varsigma_j, j = 1, 2, ..., n \\ & && \sum_{p=1}^{k} z_{i,p} = 1 \\ & && M_j \in \mathbb{N}^T, \varsigma_j \in \mathbb{N}^P, j = 1, 2, ..., n \\ & && z_{i,p} \in \{0, 1\}, i = 1, 2, ..., N, p = 1, ..., k. \end{aligned} \tag{8.8}$$

Solving the optimal problem in equation (8.8) provides a solution for the optimal path for each robot to construct the control PN. Thanks to PN simulation tools [26], different combinations of assigning the $k-$th paths to the robots are examined to find the best configurations of $z_{i,p}$ to minimise $\sum_{j=1}^{n} f(\varsigma_j)$. After this, the PN is fixed with the optimal path selection (the paths which are not selected are disabled) and the use of PN control in path planning is applied in the controlling stage.

8.2.5 Computational complexity and solution at a large scale

The computational complexity of the approach is the time needed to evaluate all combinations of assigning different paths to a robot. Assume that there is a maximum of K paths for each robot and N is the total number of robots, the evaluation of all samples runs in $O(K^N)$. As the computational complexity grows exponentially with respect to N, the algorithm will take a long time to find the optimal solution with a large number number of robots. Therefore, in order to apply the solution in a large scale, in this paper, the optimisation with a subgroup is proposed. The whole working space is divided into separated zones where the movements of robots inside each zone are synchronised by a PN. A robot travels through a sequence of zones until reaching its goal. As shown in

the example of Figure 8.8, Robot 1 ($r1$) has to pass through Zone 1, Zone 2, and Zone 3 to approach its goal. To join a zone, robots wait at the boundary of the zone until the controlled PN of the zone is free. Therefore, Robot 1 and Robot 2 ($r2$) must stop until the PN at Zone 2 completes controlling Robot 3 ($r3$) and Robot 4 ($r4$). Let D be the maximum number of robots allowed to enter the zone simultaneously. The complexity of running optimisation at each PN controller is reduced to $O(K^D)$, which is fast if D is small enough.

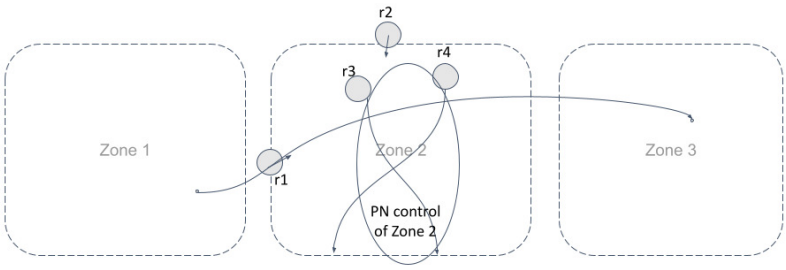


Figure 8.8: PN control by subgroups.

8.2.6 Movement control on optimal paths

The whole system is organised in a centralised manner where one master ROS node is used to create the PN model for each robot and share information with all other robots via communication channels. Once the above PN model is built, the realisation of the path planning becomes the step-by-step moving of a robot from one place to another along the generated trajectory until the robot reaches its goal. The movement of each robot is controlled by firing the enabled transitions and following a sequence of places visited by the token assigned to that robot. Each robot is linked to one token given in Line 6 of Algorithm 4. The movement controlling algorithm for a team of robots is implemented by the Algorithm 5. The PN planning mainly deals with the congestion of robots by preventing several robots to be navigated into a narrow place at the same time. The prerequisite requirement for this control with PN is that the

global planner paths are shared by the robots. However, this is not applied for humans and other moving obstacles unless they explicitly share their moving trajectories. Thus,to avoid these objects, the local path planning with DWA and dipole field is applied.

Algorithm 5 The movement control algorithm.

Input: The Petri net model (P, T, W, M_0)
Output: Step-by-step moving strategy for each robot of the team

Let $M \leftarrow M_0$
Based on the target goals, define the ending marking M_e
while $M \neq M_e$ **do** Find all enabled transitions $\dot{T} \subseteq T$
  for $t \in \dot{T}$ **do**
    Allow a robot to move if its related token is fired into a new place
    Use local obstacle avoidance while the robot is on the moving way to the next place
    Time of arrival of the token to the next place is updated by the moving time of the robot
    Update the marking M by firing t, $M \leftarrow M + W^+(t) - W^-(t)$
  end for
end while

8.2.7 Dipole field and DWA for moving obstacle avoidance

The dipole field has been introduced by the authors for obstacle avoidance with moving human or robotic agents [14]. In this method, robots and obstacles (named agents in general) are modelled by small magnets with their dipole moment vectors aligned with the moving directions of the robots and obstacles. In a consequence, two agents moving toward each other or having the same poles facing each other are pushed far away by magnetic forces. The magnetic field $\mathbf{B}$ of the dipole moment vector $\mathbf{m}$ generated by an agent is given by

$$\mathbf{B}(\mathbf{m}, \mathbf{d}) = \rho(3(\mathbf{m} \cdot \hat{\mathbf{d}})\hat{\mathbf{d}} - \mathbf{m})/d^3 \tag{8.9}$$

where $\mathbf{d}$ is the distance vector, $d = \|\mathbf{d}\|$, $\hat{\mathbf{d}} = \mathbf{d}/\|\mathbf{d}\|$, in which $||.||$ denotes the norm of the vector, and ρ is a constant. The magnetic moment $\mathbf{m}$ is designed to be aligned with the moving directions of the agent and its magnitude is proportional to the speed of the agent. An agent with the magnetic moment $\mathbf{m_j}$ within the magnetic field $\mathbf{B_k}$ of $\mathbf{m_k}$ would be affected by the force

$$\mathbf{F_{jk}} = \nabla \mathbf{m_j} \cdot \mathbf{B_k} = \rho \nabla \Big(\mathbf{m_j} \cdot \frac{3(\mathbf{m_k} \cdot \hat{\mathbf{d}})\hat{\mathbf{d}} - \mathbf{m_k}}{d^3} \Big) \tag{8.10}$$

where the gradient ∇ presents the change of potential $\mathbf{m_j} \cdot \mathbf{B_k}$ generated per unit distance, and ρ is a constant.

Dynamic window approach (DWA) was introduced by Fox et al. [21] and later developed in ROS [9] to use multiple constraints of velocity limits, of acceleration limits, and of following the predefined global path into the local path planning. The local searching space is reduced to dynamic windows in a three-step progress (Figure 8.9). Firstly, DWA considers the robot's trajectories as circular trajectories or curvatures determined by a set of translation and rotation velocities $(v_i = \dot{x}_i(t), \omega_i)$. Secondly, only admissible pairs of (v_i, ω_i) corresponding to their trajectories are considered if the robot is able to move forward without colliding with obstacles. Finally, the dynamic window limits the admissible velocities to those that the robot can safely reach to the goal in a short time with optimised accelerations.

Only the pair of (v_i^*, ω_i^*) is selected if the objective function reaches to maximum,

$$(v_i^*, \omega_i^*) = \underset{(v_i, \omega_i)}{\operatorname{argmax}} \quad F(v_i, \omega_i), \tag{8.11}$$

where $F(v_i, \omega_i) = f\Big(\sum_s q_s Q_s(v_i, \omega_i) \Big)$, $Q_s(v_i, \omega_i)$ and q_s are the cost function and the corresponding weight, and $f(\cdot)$ is an optional function used to smoothen the weighted sum of the above components. In the implementation of DWA on the ROS navigation stack[1], the objective function given by equation

[1] http://wiki.ros.org/dwa_local_planner

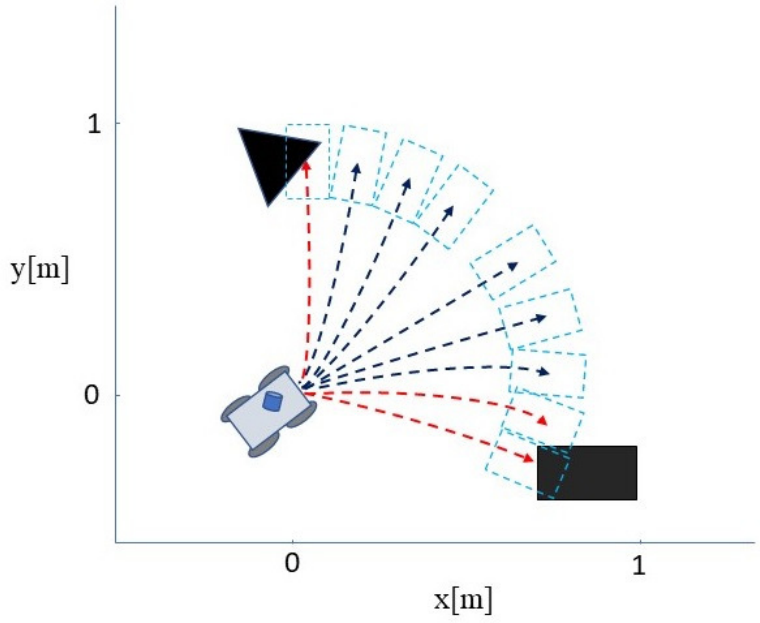


Figure 8.9: Dynamic obstacle avoidance with DWA where the feasible trajectories are blue. The read ones are leading to collisions with obstacles.

(8.12) is used to score the admissible velocities,

$$F(v_i, \omega_i) = \alpha D(v_i, \omega_i) + \beta H(v_i, \omega_i) + \gamma P(v_i, \omega_i). \tag{8.12}$$

With regards to the grid-based presentation of the cost map, $D(v_i, \omega_i)$ is the total cost of the grids passed through by the trajectory (v_i, ω_i). Meanwhile, $H(v_i, \omega_i)$ is a goal heading which is the distance from the endpoint of the trajectory to the goal, $P(v_i, \omega_i)$ is the distance from the endpoint of the trajectory to the global path, α, β and γ weight each term of the equation. The trajectory with the minimised cost $F(v_i^*, \omega_i^*)$ in (8.12) is the one with the highest possibility of avoiding static obstacles, going toward the goal, and being close to the planned path. Additionally, the dipole field has been introduced in this paper to take into account the moving directions and velocity of the obstacles to avoid the collisions with dynamic obstacles.

To integrate the dipole field into DWA, it is noted that the potential changes reflect the repulsive forces that prevent agents to collide with each other. Therefore, the new terms $\Omega_j(v_i, \omega_i)$ are added into equation (8.12) to weight the trajectories based on the possibility of collisions or to minimise the total dipole forces applied on the robots. The dipole field term is removed if $\Omega_j(v_i, \omega_j)$ does not generate repulsive forces (two objects do not move toward each other). The pair (v_i, ω_i) and its trajectory are also removed if they will cause the collisions with another robot or moving objects with regards to the velocity constraints introduced by Fiorini and Shiller [23]. It means for every neighbouring robots $j \neq i$ around robot i,

$$\|x_i(t) - x_j(t) + (\dot{x}_i(t) - \dot{x}_j(t))\tilde{t}\| \geq 2r \tag{8.13}$$

for all $\tilde{t} \in [0, \tau]$, where τ is . and for other humans and moving obstacles,

$$\|x_i(t) - o_j(t) + (\dot{x}_i(t) - \dot{o}_j(t))\tilde{t}\| \geq r + r_h. \tag{8.14}$$

8.3 Simulation and Evaluation

The whole system is implemented in the Robot Operating System (ROS) platform [24] with version Kinetic Kame in Ubuntu 16.04 on a laptop with Intel i7 CPU of 8 cores and a clock frequency of 2.90 GHz. The Gazebo simulator is used to provide realistic simulation of robots working in indoor environments. The simulated robotic agents are Husqvarna research platform (HRP)[2] mowers with an extra depth sensor, and a LIDAR laser scanner. The centralised control is performed by a master ROS node and robots in the working space share their locations and global path information through ROS messages. The PN planning module is programmed in Python with the support of the SNAKES library [26]. Meanwhile, the global path planning with the Theta* algorithm and the obstacle avoidance with dipole field on DWA are implemented in C++. In all experiments, it is assumed that all robots travel with the same speeds. Therefore, instead of using the time to analyse PN, the length of the travelling path is used as the delayed time at each transition in all simulated experiments.

8.3.1 Simulation scenarios

Two simple scenarios are simulated in this section to describe the working principle of the system.

Scenario 1. In the first scenario, the two robots are placed at opposite sides of a corridor where one of them moves from top to bottom and the other robot travels in the opposite direction (Figure 8.10). The size of the map is set to 10×10m. The global path planning returns two alternative paths for each robot where each robot has one goal location. The shortest path goes through the corridor and the other goes around the sides, outside the corridor. For both two robots described in Figure 8.10.B, Path 1 is the shortest path going through the corridor while Path 2 goes by the side of the map. The green and blue circles indicate the starting and ending positions of the robots while the red one is the intersection point if both robots choose the shortest paths.

[2] https://github.com/HusqvarnaResearch/hrp

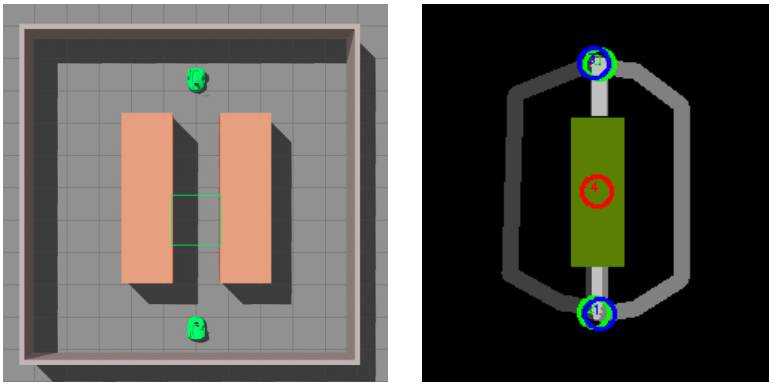
Table 8.1: The time delay at all transitions in the PN model. The time delay is equivalent to the travel distance from one place to another, so that the time unit is not necessary used in experiments.

τ_0	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7
0.84	6.45	1.14	12.36	0.64	6.51	1.53	12.59

Thus, if both robots choose the shortest path, they will meet each other at the centre of the corridor. A restricted area lying at the middle of the corridor is defined to allow only one robot to pass through. A timed PN is created for the whole system to analyse the best configurations for all robots (Figure 8.11). The delayed times at transitions are given in Table 8.1, where the delays at transition t_1 and t_5 are 6.45 and 6.51 meters, corresponding to the travelling distance of robots through the corridor.

Different combinations of pairs of paths for the robots are composed to find the best routes for the robots. If both Robot 1 and Robot 2 select to follow Path 2, the firing sequence of transitions is $\{t_3 \rightarrow t_7\}$, the total travelling time of two robots is 24.95. Similarly, if Robot 1 chooses the Path 2 and Robot 2 chooses Path 1, the corresponding firing sequence and total travelling time are $\{t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_3\}$ and 21.04 respectively. In case when Robot 1 chooses Path 1 and Robot 2 chooses Path 2, those values are $\{t_0 \rightarrow t_1 \rightarrow t_2 \rightarrow t_7\}$ and 21.02 and finally when Robot 1 chooses Path 1 and Robot 2 chooses Path 1, those are $\{t_0 \rightarrow t_4 \rightarrow t_1 \rightarrow t_5 \rightarrow t_2 \rightarrow t_6\}$ and 23.42. The best configuration achieves when Robot 1 follows Path 1 and Robot 2 is at Path 2. With the best routes selected by analysing the timed PN, the congestion does not happen in this scenario (Figure 8.13).

Scenario 2. In the second scenario (Figure 8.14), Robot 2 moves as previously while Robot 1 moves from left to right. In this scenario, the shortest path of one robot crosses that of the other at the centre. The size of the map is also 10×10m, and each robot is able to find two different paths from starting to ending points. Similar to the previous experiment, the green and blue circles mark the starting and ending location of the robots, the red circles mark the



(A) (B)

Figure 8.10: (A) The simulated working space with two robots that are placed opposite and move toward each other. (B) The global paths of the two robots.

control places. For Robot 1, starting and ending points are marked by green circle nr. 0 and blue circle nr. 1. Path 1 is the straight line from 0 to 1 through the control place nr. 7 and 8. The other path is Path 2. For Robot 2, the straight line from starting point marked with green circle nr. 2 through the control place nr. 7 and 5 to the ending position marked with blue circle nr. 3 is Path 1. The other is Path 2. Based on all the found paths, different moving scenarios are proposed to select the best scenario for the robots with the timing constraints. The delay at all transitions are depicted in Table 8.2, where the delays at transition $t_1, t_3, t_6, t_8, t_{11}, t_{13}, t_{16}$ and t_{18} are shorter than the values in Scenario 1 since the robots can pass through the crosses with a shorter time. No restricted areas are defined.

The travelling time for firing sequence $\{t_0 \rightarrow t_1 \rightarrow t_{15} \rightarrow t_{16} \rightarrow t_2 \rightarrow t_3 \rightarrow t_{17} \rightarrow t_4 \rightarrow t_{18} \rightarrow t_{19}\}$ is 35.14 when Robot 1 moves along Path 1 and Robot 2 chooses Path 2. For the sequence $\{t_{10} \rightarrow t_0 \rightarrow t_{11} \rightarrow t_1 \rightarrow t_{12} \rightarrow t_2 \rightarrow t_{13} \rightarrow t_3 \rightarrow t_{14} \rightarrow t_4\}$ which is in case that both of the robots choose to follow Path 1, the travelling cost is 20.20. If the robots select Path 2 which is

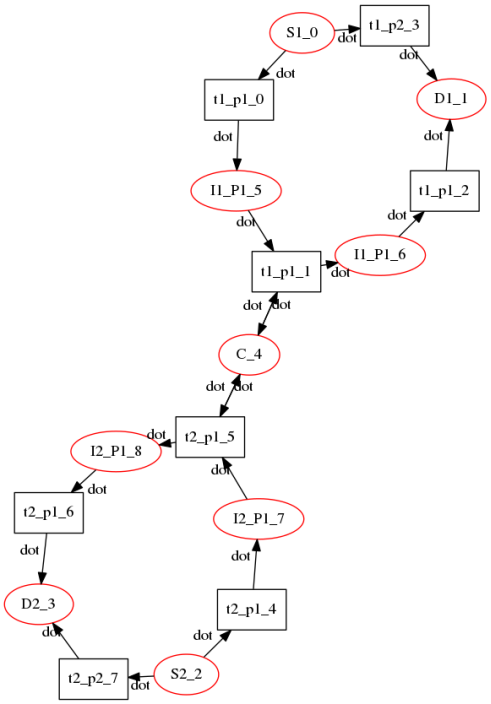


Figure 8.11: Generated PN to control the movements of two robots in the first example. The places are represented by ovals while the transitions are with rectangles. $S1\_1, S2\_2, D1\_1$ and $D2\_2$ are the source and destination positions of Robots 1 and 2. $I\{robot\_index\}\_P\{path\_index\}\_\{place\_index\}$ and $C\_\{place\_index\}$ are the intermediate places and control places of the robots respectively.

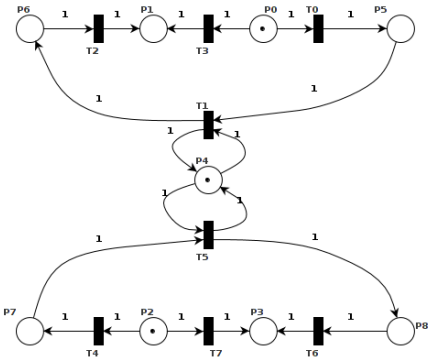
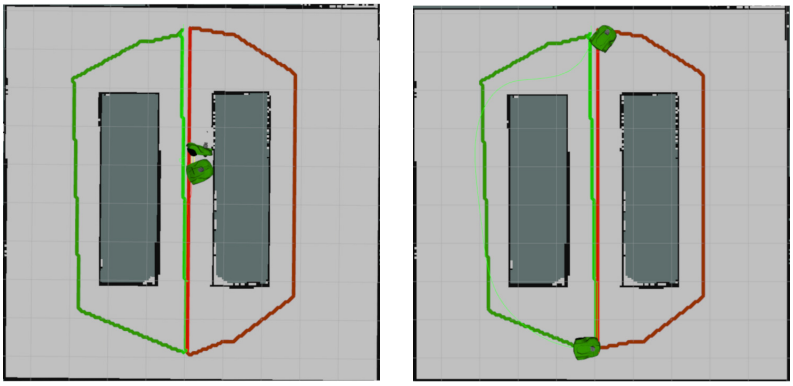


Figure 8.12: The conventional PN to visualise the generated PN model in Figure 8.11.



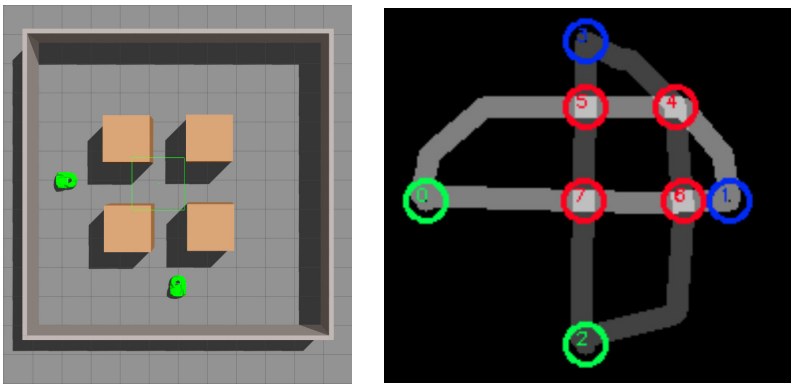
(A) (B)

Figure 8.13: (A) No PN analysis is used. Two robots collide and get blocked inside the corridor. (B) By analysing the timed PN, one robot is assigned a route going through the corridor while the other moves outside the narrow area to avoid congestion/collision. Note that the two global paths are highlighted by bold lines while the thin ones are the real trajectories.

Table 8.2: The time delay at all transitions in the PN model. The time unit is not used in experiments.

τ_0 3.09	τ_1 1.47	τ_2 1.07	τ_3 1.47	τ_4 0.69	τ_5 4.76	τ_6 1.45	τ_7 0.80
τ_8 1.52	τ_9 2.46	τ_{10} 2.73	τ_{11} 1.47	τ_{12} 0.90	τ_{13} 1.47	τ_{14} 1.19	τ_{15} 4.72
τ_{16} 1.49	τ_{17} 0.94	τ_{18} 1.52	τ_{19} 2.45				

indicated by the sequence $\{t_{15} \rightarrow t_5 \rightarrow t_{16} \rightarrow t_6 \rightarrow t_{17} \rightarrow t_7 \rightarrow t_{18} \rightarrow t_8 \rightarrow t_{19} \rightarrow t_9\}$, the total travelling time is 29.73. Finally, 46.65 is for the sequence $\{t_{10} \rightarrow t_{11} \rightarrow t_5 \rightarrow t_{12} \rightarrow t_6 \rightarrow t_{13} \rightarrow t_{14} \rightarrow t_7 \rightarrow t_8 \rightarrow t_9\}$ when Robot 1 chooses Path 2 and the other follows Path 1. With regard to those analyses, both robots select the path with the same index 1 to follow. In this case, the PN control is applied to synchronise the movements of two robots passing through the cross (Figure 10.4).



(A) (B)

Figure 8.14: (A) The simulated working space with two robots that are placed on the bottom and the left side of the map. Their shortest paths from the start to the goal are perpendicular to each other. (B) The visualisation of two global paths for each robot and the intersection of those paths.

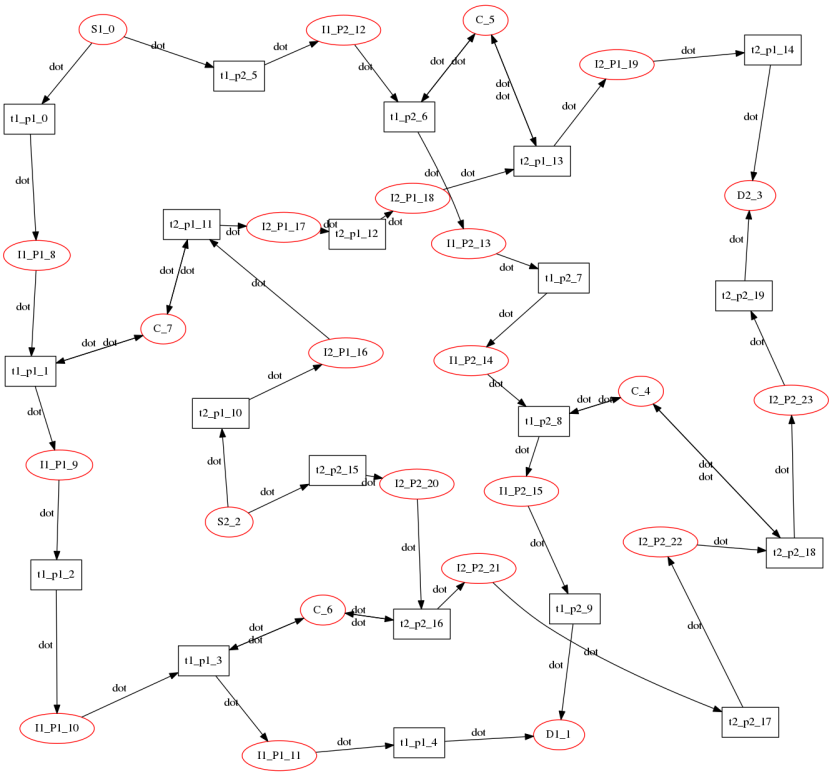


Figure 8.15: Generated PN to control the movements of two robots in the second example.

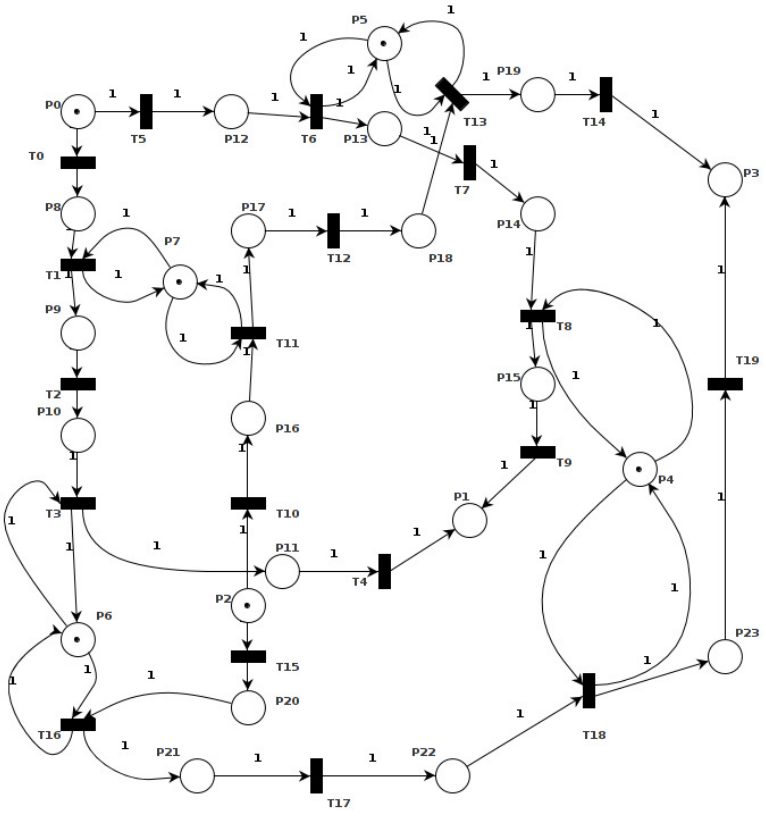
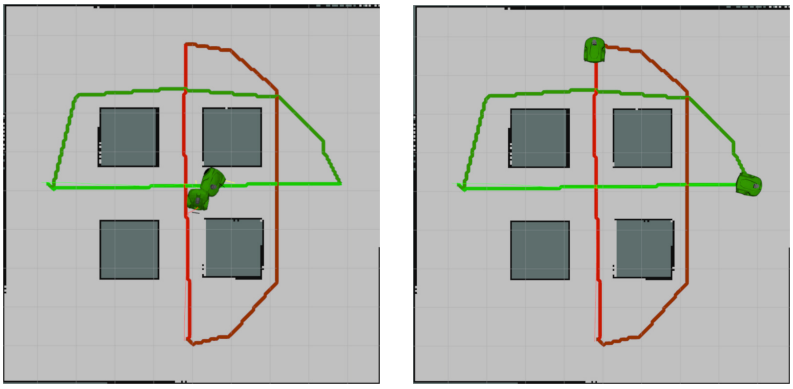


Figure 8.16: The conventional PN to visualise the generated PN model in Figure 8.15.



(A) (B)

Figure 8.17: (A) Without PN control, the two robot are not able to slow down before reaching the cross, leading to a collision. (B) With PN control, one robot stop and wait for the other passing through the cross. Two robots smoothly approach the goals with no collision. Note that the two global paths are highlighted by bold lines while the thin ones are the real trajectories.

8.3.2 Evaluation

In this section, the performance of the proposed method is verified through the evaluation of all the proposed scenarios.

Multiple robots controlled by a single Petri Net

An extensive evaluation is performed with four robots that are placed at different sides of the map (Figure 8.18). The size of the map is set to 20×20m. Similar to the previous examples, each robot is able to plan two different routes to reach its goal. PN is used to analyse the best routes and to synchronise the movements of robots. The experiment is repeated 100 times by randomly defining different goal locations. For each run the starting points are kept the same. One restricted area is established inside the corridor.

The results of three example runs are depicted in Figure 8.19. The first row describes the situation of the cross present inside a corridor. Without PN

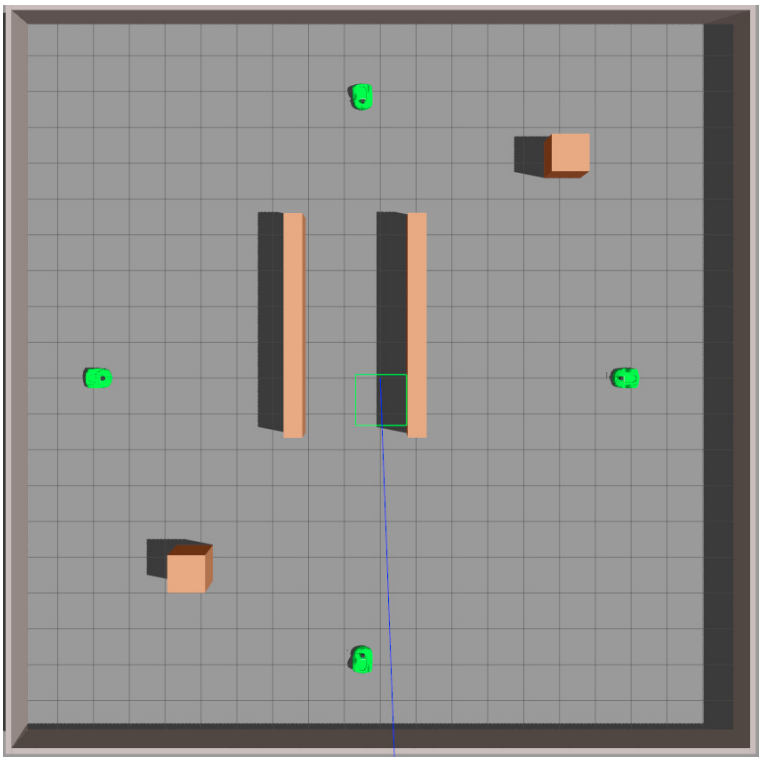


Figure 8.18: Gazebo simulation of a working space with four robots.

control, the two robots follow the shortest paths leading to the collision inside the corridor even though the two robots manage to keep on moving to reach their goals (Figure 8.19(A_1)). The result with PN control is depicted in the second row showing that no collisions occur as one robot goes outside the corridor. In Figure 8.19(A_2), there are figures expressing the case in which the cross is on an empty space. Without the PN control, a collision happens (Figure 8.19(B_2)). The PN control helps to make the moving trajectories of both robots smooth (Figure 8.19(B_2)). A comprehensive example of all paths included in a single PN model is given in the third row. There is no collision in two cases of not using (Figure 8.19(C_1)) or using the PN control (Figure 8.19(C_2)). However, in the latter, with the designed path of less intersections, a robot with the red global path is able to keep a smoother trajectory to the goal.

In Figure 8.20 the histogram of all occasions of of when the distance between two robots is less than 1.0 meters in the overall 100 trials. The results are compared with the DWA baseline algorithm on ROS [9]. The mean of those distances is 0.76m for the PN control with a standard deviation (STD) of 0.14m. Similarly, the mean and STD are 0.73m and 0.14m for the DWA-baseline algorithm. Using PN to synchronise the movements of robots, the occasions when the distance is less than 0.7 meters are reduced significantly. Note that collisions risk to happen when the distance between two robots is in the range $[0.5, 0.7]$ meters, dependent on whether they are moving toward each other in an opposite direction or they glide over each other. The summary of the number of collisions and deadlocks are given in Table 8.3. The deadlocks happens when the two robots have a strong hit that makes them stop or a robot is locked inside a corridor. There are several cases where a goal lies on the designed global path of another robot. Once a robot finishes its path and ends at the goal, it consequently stops its PN control, leading to a collision and/or even a dead lock. In these cases, two deadlocks are recorded although PN control is used.

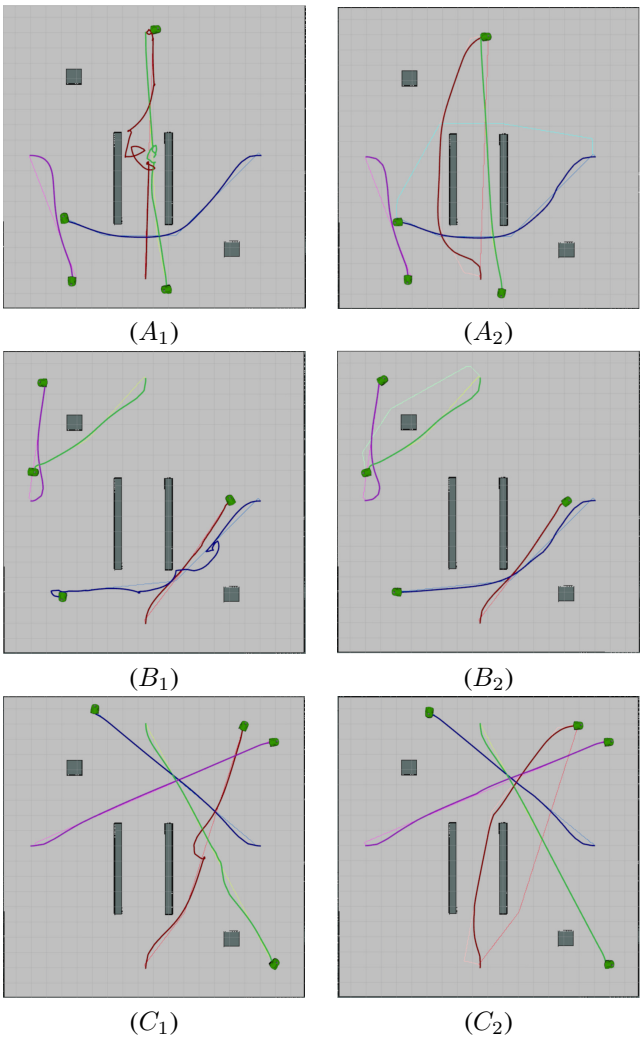


Figure 8.19: A scenario with four robots with a cross inside corridor. The trajectories of the robots without PN control $(A_1), (B_1), (C_1)$ and with PN control $(A_2), (B_2), (C_2)$. Robot 1: red, Robot 2: green, Robot 3, purple, and Robot 4: blue. Thin lines are planned paths while the thick ones are the real trajectories.

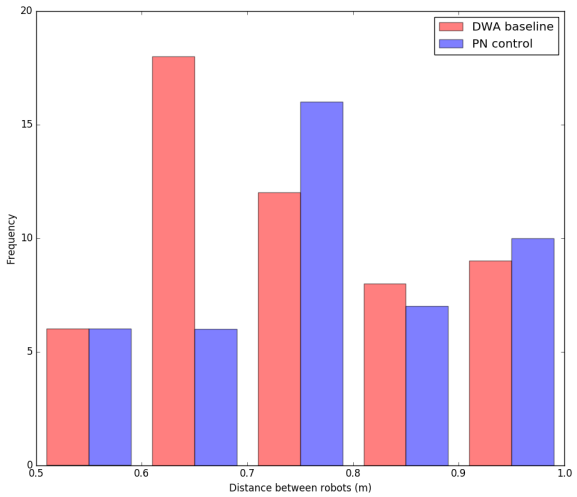


Figure 8.20: A histogram of the distances between robots less than one meter.

Table 8.3: The number of collisions/deadlocks of multiple robots with DWA baseline, with PN control (*), and with PN control when the cases of a goal on the global path of another robots are not considered (**).

	DWA Baseline	w PN Control*	w PN Control**
`Collision`	18	8	5
`Deadlock`	8	2	0

Multiple robots controlled by multiple Petri Nets

The optimisation by subgroups proposed in Section 8.2.5 shows the way to search for optimal global paths with a large number of robots in a working space. An experiment is designed in this section to evaluate the effectiveness of the proposed approach. The map with the size of 40×40m with several obstacles and narrow corridors is used. Sixteen robots are distributed equally on each side of the map. The starting positions of the robots are fixed while their goals are uniformly assigned with the separation of 3 meters. The minimum distance between a starting point and its corresponding goal is 5 meters.

As depicted in Figure 8.21, the whole working space is divided into 9 non-overlapping zones where each zone is controlled separately by one PN controller. Zone 4 takes into account only restricted areas (narrow corridors in the map) to create a PN. The maximum number of 8 robots is allowed to simultaneously enter a zone while at most three global paths are assigned for a source-destination pair. The experiment is repeated 10 times so that a total number of 160 path planning tasks are evaluated. The proposed algorithm is compared against the baseline DWA[9] and VO-based DWA path planning[38]. Table 8.4 summarises the success rate of completing the navigation tasks of the three methods. The baseline DWA mainly fails due to the collisions of robots in both empty spaces and narrow areas. The VO-based DWA has a deadlock problem when two robots meet each other in a narrow corridor. Meanwhile, the failures of the PN-based control usually happens with a single robot when it has a sharp turn and collides with a wall.

In overall, the PN-based control is superior to the other two methods on its reliability to complete the path planning tasks in this experiment. The processing time (averaged over 50 trials) of running a PN to find the optimal configuration of global paths is $\mathbf{8.7}$ **seconds**. Note that the optimisation is only required once a group of robots enters a new zone.

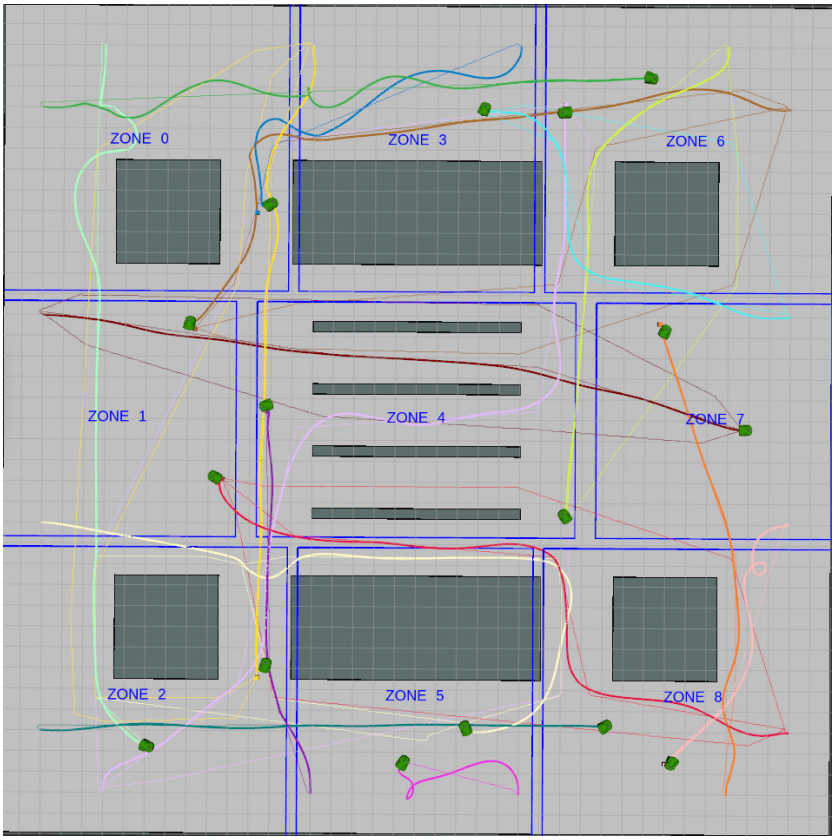


Figure 8.21: Multiple PN controllers on different zones. Note that the thin lines are the multiple planned paths and the thick ones are the actual moving trajectories.

Table 8.4: Comparisons of the proposed PN-based control with other path planning methods.

	DWA Baseline	VO-based DWA	PN Control
Success rate	82%	92%	97%

Robots and humans sharing working space

In this experiment, two humans acting in the simulated working space, which contains three robots, are added. The human actor plugin of Gazebo simulator is customised to allow control the moving trajectory of a human subject by repeated patterns (Figure 8.22). The blob-based detection using the laser scanner developed by the SPENCER project [25] is applied to track the trajectories and velocities of moving human subjects and obstacles in the working space. The same size of the map as earlier experiments of $20\text{x}20m$ is used. The experiments were repeated 5 times. For each trial, the static obstacles, start and goal positions of both robots and humans are randomised. The minimum distances between human and robot are recorded for evaluation purposes. The overall result is summarised in Table 8.5. With the combination with the dipole field on top of PN path planning, there are no collisions between robots and humans recorded.

Table 8.5: Minimum distance (in meters) between each human and robot in every trials. H1: Human 1, H2: Human 2, R1: Robot 1, R2: Robot 2, and R3: Robot 3. The **bold** numbers mark the distance between a robot and a human less than one meter. Without dipole field, there is one collision present in Trial 3 (marked with a **bold and underlined** number). Mean/STD are calculated on the distances less than 5 meters.

		T1	T2	T3	T4	T5	Mean/STD
`without Dipole Field`	H1-R1	3.38	3.52	0.63	**0.86**	3.48	2.01/1.28
	H1-R2	**0.74**	3.42	**0.88**	1.41	**0.81**	
	H1-R3	1.43	**0.90**	3.94	4.25	**0.87**	
	H2-R1	3.46	5.28	**0.69**	1.99	2.76	
	H2-R2	6.64	2.84	**<u>0.33</u>**	1.43	1.31	
	H2-R3	7.40	5.00	5.47	3.16	3.77	
`with Dipole Field`	H1-R1	1.02	3.53	**0.89**	1.03	3.84	2.26/1.12
	H1-R2	2.46	3.13	1.68	1.33	1.64	
	H1-R3	1.80	1.25	3.98	3.19	1.13	
	H2-R1	1.47	5.29	1.43	1.70	1.77	
	H2-R2	4.09	1.89	1.03	1.68	2.55	
	H2-R3	5.81	4.69	5.50	2.90	3.90	

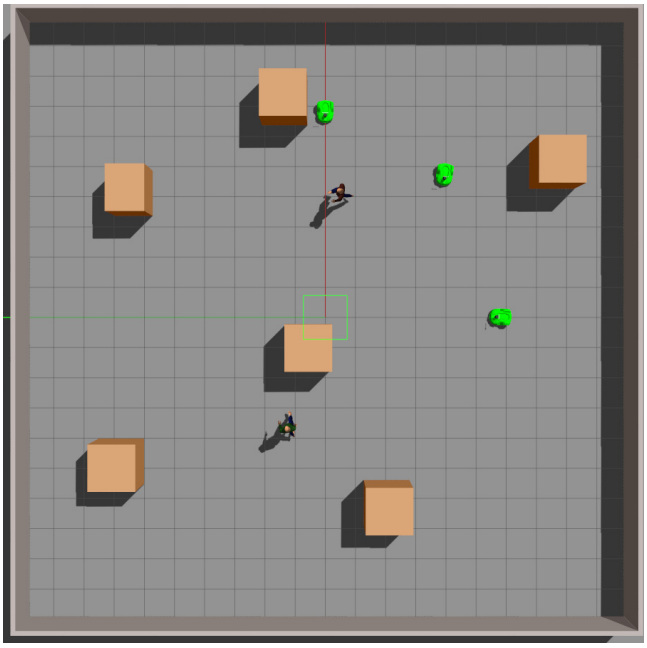


Figure 8.22: The working space with two humans and three robots simulated in Gazebo.

8.4 Conclusions and Discussion

This paper has introduced a novel path planning algorithm for multiple robots using PN in combination with obstacle avoidance with dipole field. To avoid the congestion of routing several robots into an intersection area, PN is utilised for path optimisation and then movement control of robots. Particularly, the path planning system establishes alternative routes to the goal for each robot. Due to the effectiveness PN demonstrates in analysing the travelling time from the starting position to the goal point, the optimal paths are assigned to the robots to increase the chances of a robot to reach its goal without facing dead-lock situations inside narrow areas. In the second stage when the global paths are configured with optimal selection, the PN model is applied to control the movement of robots to ensure that a robot is allowed to pass through a cross one by one. The experimental results with Gazebo simulator have revealed that the PN control is able to synchronise the movements of multiple robots passing through the intersection, which helps to both avoid dead-lock and shorten the travelling paths of the robots. Meanwhile, the dipole field implemented with DWA is able to advance the local path planning with an ability to avoid moving humans and other robots as well as uncontrolled obstacles in the shared workspace, as is also shown in simulations. Thus, these results show that the presented algorithm can improve the dependability aspect of a navigation system, based on path planning, for multiple robots system.

Some limitations of the proposed method are listed to be addressed in the future. Firstly, in the current configuration one token is assigned at the control PN place of a narrow area, which allows only a single robot to pass through. This is to minimise the possibility of a deadlock happening inside the area. If the control area is big enough, it is more optimal to allow several robots to enter it at the same time. Therefore, the correlation between the size of the region and the number of a token at the control place should also be considered as a factor to be optimised with the modelled PN. Secondly, the proposed system is mainly implemented in a centralised manner. Thus, a shift into a distributed algorithm

is considered so that the system is not so dependent on a single central server to operate. Lastly, the evaluation of the proposed approach is currently performed only as a Gazebo simulator. It is plausible to assume that an implementation in real robots will help to evaluate the proposed algorithm.

8.5 Acknowledgements

The research leading to the presented results has been undertaken within the research profile DPAC - Dependable Platform for Autonomous Systems and Control, funded by the Swedish Knowledge Foundation.

Bibliography

[1] T. S. Standley, "Finding optimal solutions to cooperative path finding problems," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI*, pp. 173-–178, July 11-15, 2010.

[2] M. Goldenberg, A. Felner, R. Sternand, G. Sharon, N. Sturtevant, R. C. Holte, and J. Schaeffer, "Enhanced partial expansion A*," *Journal of Artificial Intelligence Research*, vol. 50, pp. 141–187, 2014.

[3] M. Barer, G. Sharon, R. Stern, and A.l Felner, "Suboptimal variants of the conflict-based search algorithmfor the multi-agent pathfinding problem," in *Proceedings of the Seventh Annual Symposium on Combinatorial Search*, pp. 19—27, 2014.

[4] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proceedings of AAAI19-Thirty-Third AAAI conference on Artificial Intelligence*, pp. 7643–7650, 2019.

[5] , D. Silver, "Cooperative pathfinding," in *Proceeding of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pp. 117–122, June 01–03, 2005.

[6] G. Sharon, R. Stern, A. Felner, and N.R. Sturtevant, "Conflict based search for optimal multi-agent pathfinding", *Journal of Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[7] , J. Yu and S. M. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, pp. 1443–1449, 2013.

[8] C. D. Manning, P. Raghavan, and H. Schutze, "Introduction to information retrieval," Cambridge University Press, chapter 17, 2008.

[9] A. Filotheou, E. Tsardouliasand, A. Dimitriou, A. Symeonidis, and L. Petrou, "Quantitative and qualitative evaluation of ROS-enabled local and global planners in 2D static environments", *Journal of Intelligent and Robotic Systems*, vol. 98, pp. 567–601, 2019.

[10] M. Cristian and K. Marius, "Robot planning based on boolean specifications using Petri net models," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, 2018.

[11] Dijkstra EW, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269—271, 1959.

[12] P. E. Hart, N. J. Nilsson and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[13] Yang GZ et al. "The grand challenges of Science Robotics", *Science Robotics*, vol. 3, 2018.

[14] L. A. Trinh, M. Ekström, and B. Cürüklü, "Toward shared working space of human and robotic agents through dipole flow field for dependable path planning," *Frontiers in Neurorobotics*, vol. 12, 2018.

[15] L. A. Trinh, M. Ekström, and B. Cürüklü, "Petri net based navigation planning with dipole field and dynamic window approach for collision avoidance," in *Proceedings of the 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2019.

[16] G. Yasuda, "Discrete event behavior-based distributed architecture design for autonomous intelligent control of mobile robots with embedded Petri nets," *Advances in Chaos Theory and Intelligent Control*, Springer, vol. 37, 2016.

[17] L. Iocchi, M. T. Lazaro, L. Jeanpierre, A. I. Mouaddib, and H. Sahli, "COACHES-Cooperative autonomous robots in complex and human populated environments," *LNCS Springer*, 2015.

[18] P. E. Miyagi and L. A. M. Riascos, "Modeling and analysis of fault-tolerant systems for machining operations based on Petri nets," *Journal of Control Engineering Practice*, vol. 14, 2006.

[19] B. Lussier, A. Lampe, R. Chatila, F. Ingrand, M. O. Killijian, and D. Powell, "Fault tolerant planning: Towards dependable autonomous robots," *Research Report, LAAS-CNRS*, 2015.

[20] J. C. Fabre, M. Lauer, M. Rot, M. Amy, W. Excoffon, and M. Stoicescu, "Towards resilient computing on ROS for embedded applications," *Proceedings of the 8th European Congress on Embedded Real Time Software and Systems (ERTS)*, 2016.

[21] D. Fox, W. Burgard, and S. Thrun S, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, pp. 23-33, 1997.

[22] A. Nash, K. Danial, S. Koening, and A. Felner, "Theta*: Any angle path planning on grids," *Journal of Intelligent Robot System*, vol. 39, pp. 533-579, 2014.

[23] P Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles", *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760-772, 1998.

[24] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y, Ng, "ROS: An open-source robot operating system," in *Proceedings of the open-source software workshop*, 2009.

[25] T. Linder, S. Breuers, B. Leibe, and K. O. Arras KO, "On multi-modal people tracking from mobile platform in very crowded and dynamic environments," in *Proceedings of the IEEE international conference on robotics and automation*, pp. 5512-5519, 2016.

[26] F. Pommereau, "SNAKES: a flexible high-level Petri nets library," in *Proceedings of PETRI NETS'15, LNCS 9115*, Springer, 2015.

[27] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning", *Technical Report*, 1998.

[28] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H, Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566-580 1996.

[29] J. P. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-Body collision avoidance," *International Journal of Robotics Research*, vol. 70, pp. 3-19, 2011.

[30] B. Daman and J. P. van den Berg, "Generalized Reciprocal Collision Avoidance", *International Journal of Robotics Research*, vol. 34, no. 12, pp. 1501-1514, 2015.

[31] J. Snape, J: P. van den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacles," *IEEE Transactions on Robotics*, vol. 27, pp. 696-706, 2011.

[32] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. A. Beardsley, and R. Siegwart, "Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots," *Distributed Autonomous Robotic Systems. Springer Tracts in Advanced Robotics*, vol. 83, pp. 203-216, 2010.

[33] P. Long, W. Liu, and J. Pan, “Deep-Learned Collision Avoidance Policy for Distributed Multiagent Navigation,” *IEEE Robotics and Automation Letters*, vol. 2, pp. 656-663, 2016.

[34] M. Soulignac, “Feasible and Optimal Path Planning inn Strong Current Fields,” *IEEE Transactions on Robotics*, vol. 27, no. 1, pp. 89-98, 2011.

[35] B. Paden, M Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, “A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles,” *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33-55, 2016.

[36] E. Galceran and M. Carreras, “A survey on coverage path planning for robotics,” *Journal of Robotics and Autonomous Systems*, Elsevier, vol. 61, no. 12, pp. 1258-1276, 2013.

[37] A. Martelli, “On the complexity of admissible search algorithms,” *Artificial Intelligence*, Elsevier, vol. 8, no. 1, pp. 1-13, 1977.

[38] D. Claes and K. Tuyls, “Multi robot collision avoidance in a shared workspace,” *Autonomous Robots*, vol. 42, no 8, pp. 1749-1770, 2018.

[39] H. Kagermann, W. D. Lukas, and W. Wahlster, “Industrie 4.0: Mitdem internet der dinge auf dem weg zur 4. industriellen revolution”, *VDI Nachrichten*, vol. 13, no. 11, 2011.

Chapter 9

Paper C: Multi-Path Planning for Autonomous Navigation of Multiple Robots in a Shared Workspace with Humans

Lan Anh Trinh, Mikael Ekström, and Baran Cürüklü
Published in The 6th International Conference on Control, Automation, and Robotics, 2020.

Abstract

Path finding for multiple robots is one of most important problems in robotics when to find a way to move robots from their starting positions to reach their respective goals without collisions. However, in the case of a complex environment with the presence of humans and other unpredictable moving objects, fixing a single path to the goal may lead to a situation where there are a lot of obstacles on the planned path and the robots may fail to realise the moving plan. To address this issue, a new approach of using multiple path planning where each robot has different options to choose its path to the goal is introduced in this paper. The information about planned moving paths are shared among the robots in the working domain, combined with obstacle avoidance constraints in local ranges, and formulated as an optimisation problem. Solution of the problem leads to the optimal moving plans of robots. The effectiveness of the proposed approach is demonstrated by experimental results.

9.1 Introduction

Path planning and obstacle avoidance are important components of robotic navigation. Advances in key technologies, in combination with public acceptance, have opened the way towards allowing several autonomous robots coexist with humans in unstructured environments. This assumes autonomous navigation. One of the challenges in this regard is handling navigation failures of multiple-robots when they are operating together in a shared working space, which is also complex and cluttered. To avoid complete failures, the robots should have recovering mechanisms so that they are able to come back to their normal activities. In this context, the complete failures happen when the robots stop working and cannot finish their moving tasks. The common procedure assumes that the global path planning searches for a path, from a start to a goal, through an empty space within a map of static obstacles. The local obstacle avoidance drives the robot to follow the planned global path while taking into account possible collisions with other robots and dynamic obstacles.

Relying on a single and a fixed global plan could lead to a deadlock, or livelock, situation where a robot take a very long time to reach its goal, or will not even be able to do so. This could happen in the case of multiple robots moving in a narrow area, with respect to the the size of the robots e.g., 2x2 of a robots diameter. Since the local navigation to avoid obstacles only takes into account the collision with other robots within a close range, a robot must turn back to the configured path to be able to reach its goal. However, if two robots are routed through a very narrow area, like a corridor, and enter it through two opposite sites, the robots may face a situation where they repeat the same moving trajectories within that area again and again without ever finding the path to the goal.

Factory workshops, and other industrial spaces (also outdoor), have emerged as important cases for navigation of multiple robots. In this case the robots transport objects between different stations, thus shared the same space with humans. In the most common setup, automated guided vehicles (AGVs) are

deployed. They are configured with a predefined moving path. Due to safety reasons, the operation of an AGV is terminated when a human enters the working zones or crosses the moving trajectories of a vehicle. Yet, replacing these systems with robots that do not follow a predefined path, and have autonomous path planning, have the potential of allowing more flexible solutions, in presence of humans and other moving objects also in unfamiliar environments. During the process of controlling the paths of multiple robots, by sharing location information to each other, robots are able to avoid collisions by moving toward any open area without interfering with trajectories of others. However, the presence of humans introduces uncertainty, since robots are not able to know the intended movement of the humans. To deal with such uncertainty, the footprint i.e., the representation of human obstacles are enlarged with respect to the probability of uncertainty, or to the safety level, to prevent the collision of human with robot. A big footprint combined with a unpredictable trajectory of a human could increase the chances of blocking all feasible moves of the robot to realise the defined global moving plan.

From the described scenarios above, it is evident that, relying on a single path planning could lead to a navigation failure when there is no feasible way to implement the path due to the obstacle avoidance function. Therefore it is important for a robot to have alternative paths to reach its goal and the robot must be able to proactively switch among solutions whenever necessary to prevent deadlock situations.

In this paper, a new navigation system with multi-path planning is introduced. Each robot is able to frequently establish multiple paths from its current position to reach the defined goal. All robots in the working domain share their sets of possible planned paths to each other via a communication channel. Consequently, an optimisation problem is formulated to find the next move of the robots with respect to the constraints, which are to ensure no interference between the planned paths among different robots and no collisions between robots and other moving obstacles. In overall, the proposed multi-path planning algorithm presents an effective mechanism for fault tolerance to recover

the robot activities to handle up to some levels of failures of the navigation system.

The rest of the paper is organised as follows. Section 2 presents related works. Section 3 describes the methodology of the proposed approaches. Section 4 provides experimental results for evaluation. Finally, Section 5 concludes the paper with discussion.

9.2 Related Works

9.2.1 Multiple path planning

The multi-agent path finding (MAPF) problem has been introduced to find collision-free paths for multiple robotic agents from starting positions to their goals. A MAPF algorithm considers multiple paths for each agent and searches for a path to optimise a criteria function like a minimum total travelling distance. However, a MAPF algorithm is mainly suitable for well-defined environment without unpredictable obstacles.

For the graph-based solutions, the robots move on a connected graph from a vertex to its neighbors in one search iteration to reach their goals. A conflict happens when two robots are to occupy a single vertex at the same time. Thus, the main aim of solving the MAPF problem is to find a set of paths passing through non-conflict vertices on the defined graph. To limit unnecessary search, an extra cost function, namely sum-of-cost, like the total maximum time for all robots to reach their goals (or the cost of the paths) is introduced as an optimal condition for the search. Since the problem is non-deterministic polynomial-time (NP) hard [1], numerous approaches have chosen to seek for a close optimal solution to reduce processing time. The A*-based search uses a heuristic function to find an optimal solution among all combinations of assigning k-agent into the graph. To deal with the exponential growth of the state-space with respect to the number of robotic agents, different methods have been applied. For instance, independence detection (ID) method by Standley [6] fo-

cused on single robots and only considered a group of multiple robots jointly when necessary.

Alternative to A*, the increasing cost tree search (ICTS) [3] proposed two-layers including high-level and low-level searching where the lower is used as a goal test of the higher. Another solution different from A*, conflict based search (CBS) is introduced by Sharon et al. [4]. In CBS, agents are constrained by a triple of parameters including the agent, occupying vertex, and time step. It means that the agent at the particular time step is refused to occupy an occupied vertex. The path is found only if all agent's constraints are satisfied. The searching is completed when the paths for every agents are resolved.

Beside the above solutions, there have been suboptimal solutions for the MAPF problem. For instance, hierarchical cooperative A* (HCA*) [5] introduced a reservation table which is used to store the path assigned into an agent. The other agents will, according to their priority, search for paths not registered in the reservation table and, after the paths are found, update the table accordingly. In an improved version of HCA* like Windowed-HCA* (WHCA*) [5], the reservation table is only applied for a limited time slot, i.e. window, when the other agents are rejected to reserve to the table. Later, in the work of Bnaya and Felner [2], a dynamic window focused around conflicts and agents likely to be involved to a conflict are prioritised to be processed next. In overall, the heuristic search A* and its variants are still costly computational solutions.

There have been researches developed to reduce the running time of the search-based algorithms with rule-based algorithms. Specific rules are defined for the movement of the agents to reduce searching time. Yet, the resulted paths from the rule-based algorithms are not always optimal. Alternatively, in the work of Yu and Lavalle [7], the path planning problem for multiple agents is modelled as a network flow and the collision-free paths are found by the integer linear programming (ILP) solver.

Most of the presented solutions for the MAPF problem are based on an assumption of a working environment without the presence of humans. It is due to that the mathematics model of those works are not defined to cover both

obstacle avoidance and multiple-path planning into one combined framework. As a result, the operation of robot will be terminated as a human enters the safety regions of robots, making the solution limited to specific applications like robotics warehouse system. In the presented work, a new method of multiple path planning is proposed to consider both the human as well as other uncontrolled moving objects as factors into the path planning problem. This helps to enhance autonomous functions of robot navigation by allowing more flexibility of robots to continue working even with the presence of other robots in unfamiliar environment.

9.2.2 Collision avoidance

A field-based approach is one way to perform obstacle avoidance. In general, the field consists of a repulsive field to push the agent away from the obstacles, and an attractive field to pull the agent towards the goal. For instance, Ok et al. [8] proposed a method with an uncertainty field which is build from Voronoi diagram from the start to the goal to create the attractive field to drive the robot to the goal and the repulsive field from the robot to the obstacles.

The main issue with using this method is that the repulsive field may push the agent to reach other obstacles or statures with the attractive field. Due to this problem, the robot may be trapped into a local optimum or loose its way toward the goal.

Controlling the speed and directions of a robot is also another way to provide the robot a collision free path. Owen and Montano [10, 11] defined velocity obstacle (VO) to estimate the arrival time of moving objects to a region of collision. The acceptable velocity is the one that helps the robot to avoid collision regions. Damas and Santos-Victor [12] developed a map of forbidden velocity zones which is constructed as a limit on the velocity of the robot to avoid collision with obstacles. When the robot enters into the forbidden zones, it may adjust its speed to avoid the obstacles. In the work of Berg et al. [9], the reciprocal velocity obstacle (RVO) is introduced. In this method, the interaction of robots is modelled in both distributed and an optimal pairwise while

the other agents are assumed to continue moving with the current speed in a straight line trajectory and a function of relative velocity may be used to predict further collision. The extensions of this method are developed by Wilkie et al. [14] which is generalised for nonholonomic robots, and are improved by Berg et al. [16] by introducing the optimal reciprocal collision-avoidance (OCRA) to prevent the problem of reciprocal dances and casts. Additionally, Berg et al. [13] integrated the acceleration while Lee et al. [15] defined the footprint of the robot as an ellipse for obstacle avoidance.

Usually, to follow the global path, the preferred velocity is defined. Yet, the presence of multiple obstacles, especially non-static obstacles, usually leads to the case where no optimal velocity is found for the next moving steps, which may lead to a deadlock situation.

9.3 Multi-path Planning with Obstacle Avoidance

9.3.1 Preliminaries

In this paper, a vector is presented in bold **x**, matrix in capital and bold **X**, and a set in mathcal $\mathcal{N}$. All robotic agents and dynamic obstacles move on a free space on a 2D-plane. Assume that there are n robotic agents in the working space, denoted by $\mathcal{A} = \{i | i \in 1, 2, ..., n\}$. The position of each robotic agent i at time t is presented by a function $\mathbf{a}_i(t) \in \mathbb{R}^2$ with the correspondent velocity $\mathbf{v}_i(t) = \dot{\mathbf{a}}_i(t)$. Correspondingly, let $\mathcal{O}_i = \{j | j \in 1, 2, ..., n_i\}$ be a set of moving obstacles detected by the agent i with position $\mathbf{o}_i^j(t)$ and velocity $\mathbf{vo}_i^j(t) = \dot{\mathbf{o}}_i^j(t)$. The footprint of a robot i is modelled by a closed disk with the radius r_i. For simplicity, every function $\mathbf{x}(t)$ by time t has an equivalent representation of **x** in short. To check for the collisions among robots and moving obstacles in a local range, the concept of velocity obstacle is utilised. The velocity of a moving robot is considered to be a straight-line constant velocity within short time, leading to the position is updated by the equation

$$\mathbf{a}(t) = \mathbf{a}(t_0) + (t - t_0)\mathbf{v}, t \geq t_0, \tag{9.1}$$

where t_0 and $\mathbf{a}(t_0)$ are the current time and position of the robot at that time respectively. Given two robots with position $\mathbf{a}_A$ and $\mathbf{a}_B$, a set of relative reference velocities $\mathbf{v}_{AB} = \mathbf{v}_A - \mathbf{v}_B$ leading to the collision within time τ is given by,

$$\mathcal{VO}^{\tau}_{AB} = \{\mathbf{v}_{AB} | \forall t \in [0, \tau], \|\mathbf{a}_A - \mathbf{a}_B + t\mathbf{v}_{AB}\| \leq r_A + r_B\}. \tag{9.2}$$

This velocity obstacle $\mathcal{VO}^{\tau}_{AB}$ is visualised by a truncated cone and approximated by the non-convex space formulated by three half planes (Fig. 9.1). Once the velocities and positions of moving obstacles are detected by a robot, the pairwise collisions between a robot and an obstacle is modelled and checked by the similar way. Yet, the moving obstacles may be seen by one one robot but not others, therefore, the collision checking between robot-moving obstacle is only available within local regions.

Meanwhile, the global map of a set of static obstacles $\mathcal{M}$ is presented by a binary image. To take into account the size of robot to avoid collisions with obstacles, the global map image is usually dilated by the radius of the robots' footprint. Let $\mathcal{M}(r)$ be a map with dilated obstacles of the radius r. The radius is usually set by the maximum radius r_{max} among different robots' footprint.

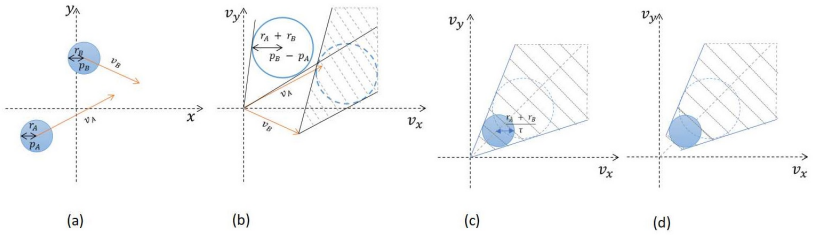


Figure 9.1: Formulation of velocity obstacles. (a) The workspace configuration of the two robots R_A and R_B with their velocities v_B and v_B respectively. (b) The translation into velocity space and the resulting VO for robot R_A. (c) The VO of an obstacle is truncated at $\tau = 2$. (d) The approximating of truncated VO.

9.3.2 Problem formulation

From the current position $\mathbf{a}_i(t)$ of the robot i, there are available p_i paths to its goal. Those paths can be established by running a random-related algorithm, e.g. rapidly exploring random tree (RRT), several (multiple) trials or by using different path finding algorithms, or even by using manual inputs from users. In this work, the any-angle searching with Theta* [19] is utilised to generate paths by using sequential inserting a set of found paths (the thickness of the path is dilated by the radius of the robot) into an obstacle map. By this way, the next found path will not overlap with the previous one. The any-angle searching Theta* is chosen instead of using A* or Djikstra's algorithms because Theta* is able to provide the optimal path with few turns and in the form of a set of line segments that reduces the changes in orientations to save energy by maintaining a constant moving speed and orientation. Also, by this way, it is convenient to find the intersections of two paths and define constraints for potential collision areas.

There is a preferred velocity of a robot defined on each path in such a way that the velocity remains constant along the path and smoothly decreases when the robot approaches to its goal. Let $\mathcal{V}_i = \{k|k \in 1, 2, ..., p_i\}$ be a set of the available paths for robot i, p_i be the number of paths, $\mathcal{P}_i = \{\bar{\mathbf{v}}_i^1, \bar{\mathbf{v}}_i^2, ..., \bar{\mathbf{v}}_i^{p_i}\}$ be a set of preferred velocities on each path. The control velocity $\mathbf{v}_i$ to determine the next move of the robot is set to be close to one of the preferred velocities as only one path is chosen among $\mathcal{V}_i$. Let $\mathbf{z}_i = [z_i^1, z_i^2, ..., z_i^{p_i}]^T$ be the binary vector to select the path, $z_i^k \in \{0, 1\}$. The optimisation cost function $C_i(\mathbf{v}_i, \mathbf{z}_i)$ is defined as follows:

$$\begin{aligned} C_i(\mathbf{v}_i, \mathbf{z}_i) &= \|\mathbf{v}_i - \sum_{k=1}^{p_i} \bar{w}_i^k z_i^k \bar{\mathbf{v}}_i^k\|^2 + \sum_{k=1}^{p_i} z_i^k \bar{s}_i^k \\ \text{s. t.} \quad & \sum_{k=1}^{p_i} z_i^k = 1 \end{aligned} \tag{9.3}$$

where $\bar{w}_i^k$ and $\bar{s}_i^k$ are the weight and the travelled lengths of the path. Without

further constraints, minimising $C_i(\mathbf{v}_i, \mathbf{z}_i)$ leads to the selection of the shortest path among candidates.

The joint optimisation function for all robots to find their optimal paths and velocities is expressed by,

$$\begin{aligned} C(\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n, \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n) &= \sum_{i=1}^{n} C_i(\mathbf{v}_i, \mathbf{z}_i) \\ \text{s. t.} \quad \sum_{k=1}^{p_i} z_i^k &= 1, \forall i \in [1, n]. \end{aligned} \tag{9.4}$$

Along with the cost function, a set of constraints are defined to find collision-free paths for robots considering that they have different options to chose their paths and also they need to avoid any dynamic obstacles on their moving ways.

Multi-path conflict-free constraints

Assume that two robots A and B are configured with multiple paths to goals (Fig. 9.2). However, some specific combinations of the path for the two robots could lead to a potential collision or deadlock. For instance, if robot A and B selects the path *p_2A* and *p_2B* for their moving plans, there exists an overlapping segment of the paths where the two robots may meet with each other. In the case the overlapping segment is bounded by a narrow area surrounding with static obstacles, there is a high possibility that the two robots are getting stuck inside the region. A set of constraints are used to penalise such combinations,

$$\begin{aligned} \mathcal{CF} = \{ z_i^k + z_j^l \leq 1 | \forall i, j \in \mathcal{A}, k \in \mathcal{V}_i, l \in \mathcal{V}_j, \\ \text{if there is a potential conflict when robot } i \\ \text{chooses the path } k, \text{ and } j \text{ chooses } l \}. \end{aligned} \tag{9.5}$$

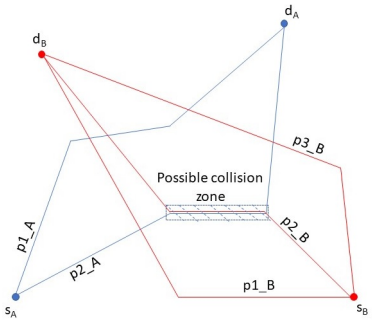


Figure 9.2: An example of creating multiple paths for two moving robots with a possible collision zone.

Moving obstacle avoidance constraints

According to the definition of VO (Section 9.3.1), the collision between robot A and another robot or a moving obstacle B is avoided if $\mathbf{v}_A - \mathbf{v}_B \notin \mathcal{VO}^{\tau}_{AB}$. This non-convex constraint $R^2 \setminus VO^{\tau}_{AB}$ is approximated by three linear constraints $\mathbf{n}^l_{AB}.\mathbf{v}_{AB} \leq b^l_{AB}$, with $l \in 1, 2, 3$

$$\begin{aligned} \begin{bmatrix} \cos(\alpha+\beta) \\ \sin(\alpha+\beta) \end{bmatrix} \mathbf{v}_{AB} &\leq 0, \\ \begin{bmatrix} \cos(\alpha-\beta) \\ \sin(\alpha-\beta) \end{bmatrix} \mathbf{v}_{AB} &\leq 0, \\ -\frac{\mathbf{p}_{AB}}{p_{AB}}.\mathbf{v}_{AB} &\leq \frac{p_{AB} - \bar{r}_{A+B}}{\tau}, \end{aligned} \tag{9.6}$$

in which $\mathbf{p}_{AB} = \mathbf{a}_A - \mathbf{a}_B$, $\bar{r}_{A+B} = r_A + r_B$, $p_{AB} = \|\mathbf{p}_{AB}\|$, $\alpha = \arctan 2(-\mathbf{p}_{AB})$, and $\beta = \arccos(\bar{r}_{A+B}/p_{AB})$. The first and second constraints are to realise the right and left side of avoidance. The last constraint makes sure that there are no collision up to up to $\tilde{t} = \tau$.

In the work of Mora et. al [18], this non-convex constraint is added into the optimisation problem by introducing extra binary variables to select one of these linear constraints to apply. However, the number of binary variables

increases rapidly with respect to the number of agents. To avoid doing so, one of the approaches used in this work is to add only one of the three linear constraints where the constraint l is selected based on the current velocity of robots at the current time t_{curr},

$$l^* =_l \ \mathbf{n}^l_{AB}.(\mathbf{v}_A(t_{curr}) - \mathbf{v}_B(t_{curr})) - b^l_{AB}. \tag{9.7}$$

Static obstacle avoidance

Since the static obstacles can be treated as moving obstacles with zero velocities, the static obstacle avoidance can be addressed using VO as presented in Section 9.3.2. However, as the combined VO areas are proportional to the size and the number of obstacles in the global map, adding many static obstacle avoidance constraints may lead to deadlock situations where the optimisation of the problem will not be able to find a feasible velocity. Therefore, in this work, the static obstacle avoidance is handled with dynamic window approach (DWA), that is described further in Section 9.4.1.

Finally, the overall optimisation problem is formulated, in which the optimal control velocities and selected global paths $[\mathbf{v}^*_{1:n}, \mathbf{z}^*_{1:n}] = [\mathbf{v}^*_1, \mathbf{v}^*_2, ..., \mathbf{v}^*_n, \mathbf{z}^*_1, \mathbf{z}^*_2, ..., \mathbf{z}^*_n]$ are estimated in a joint manner,

$$\begin{aligned}
[\mathbf{v}^*_{1:n}, \mathbf{z}^*_{1:n}] &= \underset{[\mathbf{v}_{1:n}, \mathbf{z}_{1:n}]}{\operatorname{argmin}} C(\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n, \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n) \\
\text{s. t.} \quad & \sum_{k=1}^{p_i} z^k_i = 1, \forall i \in [1, n] \\
& \mathcal{CF} \quad \text{(Constraint 1)} \\
& \mathbf{v}_i, \mathbf{v}_j \notin \mathcal{VO}^{\tau}_{ij}, \forall i, j \in \mathcal{A} \quad \text{(Constraint 2)} \\
& \mathbf{v}_i \notin \mathcal{VO}^{\tau}_{io_j}, \forall i \in \mathcal{A}, j \in \mathcal{O}_i.
\end{aligned} \tag{9.8}$$

9.4 Experiments

9.4.1 ROS-based implementation

The overall system presented in this work is implemented with a well known platform for robots, robot operating system (ROS), with the specific version of Kinetic Kame installed on Ubuntu 16.04. The evaluation is performed with the comprehensive Gazebo simulator and robotic mowers developed based on Husqvarna research platform (HRP). Those simulated robots are mounted with an extra RGB camera, a depth sensor, and a LIDAR laser scanner. As the system is operated in a centralised manner, a ROS node is designed as a server to collect information about planned paths, position updates, and velocities of all robots as well as moving obstacles detected by the robots in the working domains. The optimal velocities are computed at the ROS centre node and are sent back to the robots to control their movements. Human objects are modelled as actors in Gazebo simulator with either a repeated predefined trajectory or a random trajectory. The overall optimisation formulation for the whole system is a mixed quadratic integer programming (MQIP) problem, and the solution for the problem is calculated by IBM CPLEX solver.

To apply the estimated optimal velocities to control the movements of robots, the DWA method is used. The DWA is a commonly sampling-based approach that allows to generate a set of possible trajectories of a robot in a short time slot based on feasible velocities and limited accelerations. The trajectories are scored with regards to the distance between the robot and obstacles, the distance to reach the goal, or the deviation from the global path. The trajectories leading to collisions with static obstacles on the map are removed from consideration. In this work, to utilise DWA to realise the estimated controlled velocities for the robots, the mean velocity on each trajectory is approximated by dividing the distance between the start and end of the trajectory with the travelling time. The DWA scoring function therefore aims to find the trajectory that minimises the differences between the trajectory's velocity with the targeted velocity. In this way, both static as well as dynamic objects are con-

sidered when optimising the possible path for each robot.

9.4.2 Two robots crossing narrow corridor scenario

A typical scenario of two robots crossing a narrow corridor is evaluated in this section to demonstrate how multi-path planning is used to address the congestion problem. Two HRP robots are located at two different sides of the corridor. By applying multiple path planning, each robot has found a set of two possible paths from its starting position to its goal (Fig. 9.3). The two red paths are the paths found by the robot 1, and the green ones are for the other. The yellow paths are the actual trajectories of the two robots after they have received optimal velocity control from the centre ROS node.

If the first robot chooses to navigate through the corridor, the other one will choose the other path in order to avoid congestion in the corridor although this path is longer than its optimal route.

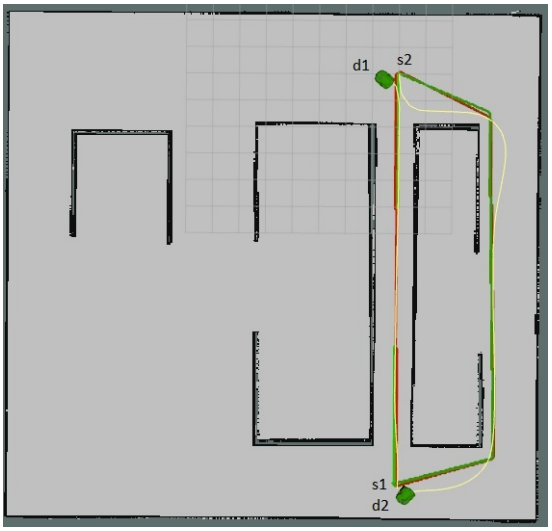


Figure 9.3: Multiple planned paths of two moving robots to reduce the risks of collisions.

9.4.3 Scenario with robots/humans together

A simulated working space with several narrow corridors as depicted in Fig. 9.4 is used to further evaluate the system on congestion and obstacle avoidance. Three HRP robots are fixed with starting points but randomly assigned goals. In a half of experiments, two robots are arranged on the two sides (top and bottom) of the map (Fig. 9.4) and move from one side to the other. Two human actors are added with predefined moving trajectories. The experiments has been repeated 10 times and the proposed algorithm is compared with DWA and DWA+VOs [17]. A collision happens if the distance between robot/robot or robot/human is less than 0.5 meters regarding the size of robots. The results (Table 9.1) show that the proposed algorithm is superior to previous works with respect to ability to avoid collisions/dead-locks. Outside the narrow corridors, the optimal control helps them to avoid collisions with humans (The minimum distance between robots and humans in all experiments is 0.51 meters when robots are controlled by the proposed navigation algorithm). This is shown by the changing directions on the moving trajectories of different robots (Fig. 9.5). It is noted that the human actors are only visualised on the Gazebo simulator.

Table 9.1: Minimum distance among robots (in meters), number of collisions, and number of dead-locks over 10 trials.

	Distance	Collisions	Dead-locks
Multi-path planning	0.56	0	0
DWA	0.21	3	5
DWA + VOs [17]	0.57	0	3

9.5 Conclusions

This paper presents a novel multiple path planning approach, which can deal with an uncertain and dynamic environment containing non-static objects such

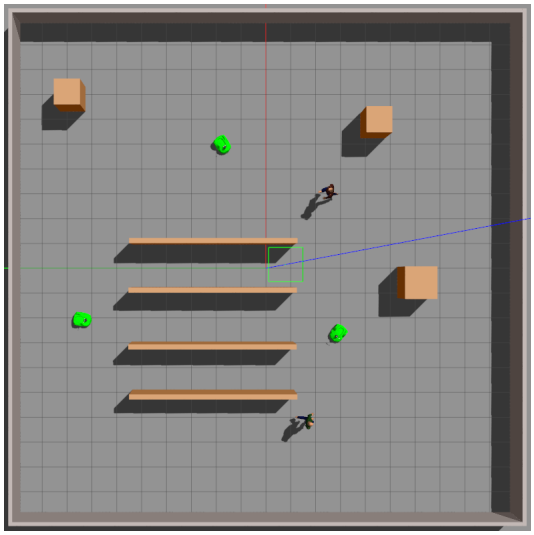


Figure 9.4: The simulated working space with three robots and humans.

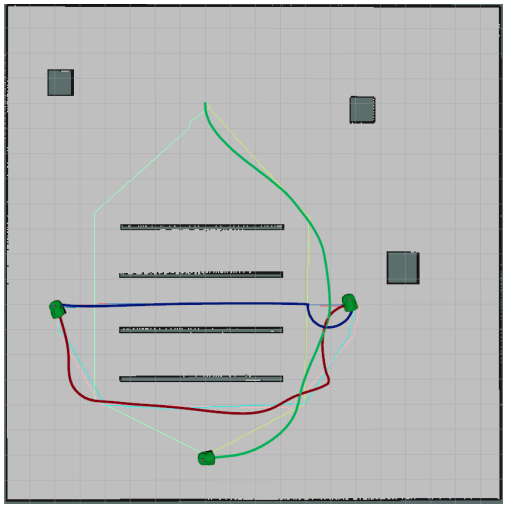


Figure 9.5: Moving trajectories of robots. The thin trajectories are the two planned path while the bold ones are the actual moving trajectories of robots.

as humans and robots. The presented method introduces effective means of a global planner for avoiding deadlock situations as well as to overcome the risk of congestion when multiple robots are navigated through, relative to the robots, a narrow area. The combination of VO-based method and common DWA planner allows to extract the velocities of both robots and moving obstacles. Moreover, the velocities of the robots are transferred into an optimisation problem to improve the performance of controlling the robots' movements. In addition, the ROS based communication channel allows the robots to negotiate between the different possibilities to have collision-free path solutions, and also to allow continuous updates of the positions of each obstacle in the environment used in calculating new possible paths. The evaluations in the Gazebo simulator has proved that the proposed approach with multiple path planning is effective, safe and promising for an autonomous robot team. In the future, the decentralised movement control unit will be investigated to reduce the dependence of the planning algorithms on communication infrastructure. Also the method can be improved by applying a delay to be lower the energy consumption of robots. The robot may, instead of choosing the longer path, wait for others to follow the shortest path to reach its goal. Finally, an extensive evaluation with real robots will be planned.

Acknowledgement

The research leading to the presented results has been undertaken within the research profile DPAC - Dependable Platform for Autonomous Systems and Control, funded by the Swedish Knowledge Foundation.

Bibliography

[1] J. Yu and S.-M. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, pp. 1443-1449, 2013.

[2] Z. Bnaya and A. Felner, "Conflict-oriented windowed hierarchical cooperative A*," In *Proceddings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Hong Kong, China), pp. 3743-3748, 31 May-7 June, 2014.

[3] G. Sharon, R. Stern, M. Goldenberg, and A. Felner, "The increasing cost tree search for optimal multi-agent pathfinding," *Journal of Artificial Intelligence*, vol. 195, pp. 470-495, 2013.

[4] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict based search for optimal multi-agent pathfinding," *Journal of Artificial Intelligence*, vol. 219, pp. 40-66, 2015.

[5] D. Silver, "Cooperative pathfinding," in *Proceeding of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pp. 117-122, June 01-03, 2005.

[6] T. S. Standley, "Finding optimal solutions to cooperative pathfinding problems," in *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pp. 173-178, 2010.

[7] J. Yu and S. M. LaValle, "Planning optimal paths for multiple robots on graphs," in *Proceeding of the IEEE International Conference on Robotics and Automation (ICRA)*, (Karlsruhe, Germany), pp. 3612-3617, May 6–10, 2013.

[8] K. Ok, S. Ansari, B. Gallagher, W. Sica, F. Dellaert, and M. Stilman, "Path Planning with uncertainty: Voronoi uncertainty fields," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Karlsruhe, Germany), pp. 4581-4586, May 06-10, 2013.

[9] J. V. D. Berg, M. Lin, and D. Manocha, " Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proceedings of the IEEE Interna-*

tional Conference on Robotics and Automation (ICRA), (Pasadena, CA, USA), pp. 1928-1935, May 19-23, 2008.

[10] E. Owen and L. Montano, "Motion planning in dynamic environments using the velocity space," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Edmonton, Canada), pp. 2833-2838, August 2-6, 2005.

[11] E. Owen, and L. Montano, "A robocentric motion planner for dynamic environments using the velocity space," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Beijing, China), pp. 4368-4374, October 9-15, 2006.

[12] B. Damas, and J. Santos-Victor, "Avoiding moving obstacles: The forbidden velocity map," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (St. Louis, USA), pp. 4393-4398, October 11-15, 2009.

[13] J. V. D. Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), pp. 3475-3482, 2011, .

[14] D. Wilkie, J. V. D. Berg, and D. Manocha. "Generalized velocity obstacles," in *Proceedings of the IEEE/RSJ International Conference on In-*

telligent Robots and Systems (IROS), (St. Louis, USA), pp. 5573-5578, October 11–15", 2009.

[15] B. H. Lee, J. D. Jein, and J. H. Oh, "Velocity obstacles based local collision avoidance for holonomic elliptic robot," *Journal of Autonomous Robots*, vol. 41, pp 1347-1363, 2017.

[16] J. V. D. Berg, S.J. Guy, M. Lin, and D. Manocha. "Reciprocal n-body collision avoidance," *International Symposium on Robotics Research*, pp. 3-19, 2009.

[17] D. Claes, and K. Tuyls, " Multi robot collision avoidance in a shared workspace," *Autonomous Robots*, Springer, 2018.

[18] J. A-Mora, P. Beardsley, and R. Siegwart, "Cooperative collision avoidance for nonholonomic robots," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 404-420, April 2018.

[19] A. Nash, K. Danial, S. Koening, and A. Felner, "Theta*: Any angle path planning on grids," *Journal of Intelligent Robot System*, vol. 39, pp. 533-579, 2010.

Chapter 10

Paper D: Decentralised Multi-Robot Path Planning with Obstacle Avoidance and Congestion Control Constraints

Lan Anh Trinh, Mikael Ekström, and Baran Cürüklü
Submitted to IEEE Access.

Abstract

The tremendous development in robotics, especially with autonomous controls, has brought advantages in industrial production and daily life applications. This paper presents a framework to tackle the path planning, one of the core component of the robotic system, for multiple robots. The proposed approach is modelled by an optimisation problem with a quadratic utilisation to minimise the travelling path length and to generate smoothly moving trajectories with regularisation. Two additional constraints are introduced. The congestion constraint aims to reduce the number of crosses on a specific path. By avoiding the path with heavy traffic jams, the robots proactively avoid the possibility of congestion, which could lead to a collision or a deadlock to prevent them reaching their goals. The other constraint deals with obstacle avoidance when several robots meet each other at an instance of time. As the selection of multiple paths is allowed, the whole problem is formed as a mixed integer quadratic programming with integer variables to select suitable moving paths for each robot. To deal with the high complexity of the problem with respect to the number of robots, robots in the working space shares their computational resources in a decentralised manner to search for the solution of the problem. This is also to enhance the reliability of the system to reduce all dependency on one central node. Extensive experiments have been performed to evaluate the efficiency of the proposed solution.

10.1 Introduction

Navigation whose the main aim is to find a clear path from a starting point to a goal without inference with any static or moving obstacles has been identified as one of the top grand challenges to create an autonomous control for a robot system [1]. The complexity of the problem is high due to the vast combinations of environments and situations which the robot is facing while moving on its way. A solution for this problem using machine learning requires enormous data to cover every cases, so that even effective learning techniques like deep learning are still having limitations of handling untrained scenarios. A basic form of artificial intelligence with a human-designed algorithm therefore plays an important role to help an autonomous robot system to overcome the boundary of the data used to train the system. Another advantage is that such an algorithm provides a transparent solution to explain the decision-making process, which is useful for the analysis, evaluation, and checking of the system's dependability.

The earliest human-designed approaches for the navigation problem started with searching algorithm. A family of A* algorithms [2] uses a heuristic function to seek for a suboptimal solution and is still the best solutions to address the path finding in many situations [3]. With regards to the changes of environment, D* (Dynamic A*) [4] and its light version [5] only updates the path based on the differences from the map. The input into A* and its variant is in the form of a graph of a dense grid or a visibility graph to mark available spaces to move and regions occupied by obstacles. In most of cases, the graph is fully constructed before the searching started. In another approach, exploring the environment is performed in parallel while seeking for the destination. Rapidly-exploring random trees (RRTs) [6] generates samples of the path until reaching the goal. In overall, the way to construct a complete path from a staring point to a goal works effectively for static obstacles. To handle moving obstacles, the temporal dimension as well as dynamics of the obstacles like velocity or acceleration need to be taken into account. This increases the

searching space and the problem is even harder considering some uncertainty to estimate the exact location of the object.

As yet, the obstacle avoidance is handled separately after the global path is established. The most common way to handle local obstacles is sampling forward trajectories of a robot based on its current locations and dynamics information [7]. The trajectories are weighted based on the closeness to the preferred global path and the interference with any objects in an obstacle map. However, this method aims to prevent collisions in a passive manner, meaning that the velocity of moving obstacles are not taken into account. Another proposed solution with velocity-obstacle (VO) [8] defines prohibited regions in a velocity space where a velocity of a robot should be outside to avoid collisions with any moving obstacles. A light computational solution with linear programming optimisation was developed with the optimal reciprocal collision-avoidance (ORCA) [9] to estimate optimal velocity with respect to a set of VOs. To deal with the return of velocity back to its previous value before escaping collisions, the hybrid reciprocal velocity obstacle (HRVO) [10] estimates the velocity on one side of the half plane of all VOs. A generalised velocity obstacle (GVO) [11] extends VO to cover different types or robots. However, to handle static obstacles, they need to be presented in velocity space, correspondingly increasing the complexity of the navigation algorithm to deal with a highly clustered map, especially when the shape of the objects must be taken into account. Executing VO-based navigation algorithm on a real map could lead to a deadlock where a robot get stuck into a place or a live-lock where a robot just goes around and takes very long time to finish its path. This problem happens because of the focuses of the obstacle avoidance algorithm on local information to handle collisions. After the global paths are assigned for each robot, all robots only follows their single path without recognising that the path can lead to a congestion in a narrow area. Therefore, a search for global path should include all robots so that each robot is able to choose a less crowded way to go.

The approaches with multi-agent path finding (MAPF) [12] include multi-

ple robots in the searches for their paths to goals without conflicts and optimise a utility function such as the total travelling lengths of robots. As the path finding is performed on a graph, a conflict happens whenever two robots occupy a same vertex or link at an instance of time. The complexity of a MAPF problem is NP-hard in general, meaning that none algorithm is proven to return solutions in polynomial time. A suboptimal solution has been found by a family of A*-based MAPF algorithms. For instance, conflict-based search (CBS) [13] generates a tree search starting with a root node with all shortest paths of robots to their goals. The tree search is expanded with child nodes to avoid collisions with constraints and then refine global paths accordingly. Improved CBS like cooperative A* [14] or priority-based search [15] prioritise the order of robots based on the crossing number of the shortest paths of a robot with those of others. Another way to tackle MAPF is to relax it into integer linear programming with the time-expanded network [16], which models the transitions of robots in graph from one vertex to the next with respect to time. However, MAPF solvers cope with fixed trajectories of robots where their arrival time to reach a vertex or a link is well predefined. Most of those solvers skip the dynamics of robots, which constrain the movements of robots so that they may slip from their trajectories due to inertia and the time reaching a target is not always predictable.

In overall, the limitations of aforementioned works are described in one of either two cases: (i) The lacking of multiple path selection leads to a dead-lock when two or more robots enter a small area which does not fit them; or (ii) Multiple ways for robots are taken into account but do not handle unpredictable or unexpected moving robots and obstacles. In order to address such problems, this paper proposes a multi-path planning for multi-robots with obstacle avoidance and congestion control constraints together. The collisions with obstacles are avoided by defining a set of constraints in a VO space. Meanwhile, the congestion control constrains the number of robots allowed to going through a place. The whole framework is formulated as a mixed-integer quadratic programming (MIQP) problem, which is solved by the operator splitting quadratic

program (OSQP) [17] in a decentralised manner. The contributions of the paper are three folds: (i) the allowance of multiple path selection helps to mitigate deadlocks, (ii) the integration of obstacle avoidance into multi-agent path planning, and (iii) the implementation of the solution in decentralised manner to make it applicable to multiple mobile robots.

The rest of this paper is presented as follows. The mathematical model of the proposed framework is formulated in Section 10.2. Continuously, the decentralised optimisation with OSQP to find an optimal moving path and velocity for each robot is described in Section 10.3 with evaluations on extensive experiments given in Section 10.4. The paper is concluded with discussions in Section 10.5.

10.2 Problems

10.2.1 Problem statement

The navigation problem to be addressed in this paper is to find the way to drive multiple robots to their goals while minimising an objective function such as the total travelling path or travelling time of all robots. Besides, robots need to avoid collisions with static obstacles from environments and mutual collisions with each other.

Preliminaries. It is assumed that robots are moving on 2D-plane where the location of a moving object is presented by a vector $\mathbf{x}(t) \in \mathbb{R}^2$ (or $\mathbf{x}$ in short) which shows the changes of 2D-coordinate of the object over time t. The matrix is capital and bold $\mathbf{X}$, while a set is shown by a calligraphic letter $\mathcal{X}$.

Robots. A set of N robotic agents $\mathcal{A} = \{a_1, a_2, ..., a_N\}$ are coped in this work. All robots are considered identical with a unit-circle model (a circle with the radius r_a). The position and the velocity of a robot i are expressed by $\mathbf{a}_i(t)$ and $\dot{\mathbf{a}}_i(t)$ respectively.

Footprints. The footprint of each robot is denoted by a circle $\mathcal{F}_i(\mathbf{x}_i(t)) \subset$

$\mathbb{R}^2$ and, which is a close disk at position $\mathbf{x}_i(t)$ and radius r.

Map and static obstacles. A binary grid map $\mathcal{M}$ in 2D is used to describe the working environment. The black pixels are the empty areas, while a set of white pixels $\mathcal{S}$ are static obstacles. Taking into account the size of a robot, $\mathcal{S}$ is dilated by the radius r to have $\mathcal{S}_r$ to allow a robot to move freely in $\mathcal{M} \setminus \mathcal{S}_r$.

The main problem to be addressed in this work is to seek for the optimal paths for every robots to reach their goals while avoiding any collisions with static and moving obstacles on their moving way. Let t_i^0 and t_i^1 be the starting and ending time for a robot i to travel along its path. The reaching goal statement is expressed by

$$\mathcal{F}_i(a_i(t_i^1)) \cap \mathcal{G}_i \neq \emptyset, \tag{10.1}$$

where $\mathcal{G}_i$ is the close disk at the goal with radius r_g. The constraints to prevent mutual collisions among robots and between robots and static obstacles are given by,

$$\mathcal{F}_i(a_i(t)) \cap \left(\bigcup_{j \in \{1,2,\ldots,N\}, j \neq i} \mathcal{F}_j(a_j(t)) \cup \mathcal{S}_r \right) = \emptyset, \qquad \forall t \in [t_i^0, t_i^1]. \tag{10.2}$$

10.2.2 Obstacle avoidance constraint

The constraint to prevent collisions of robots with obstacles is defined with VO [8]. Assume that within a short duration of time, a robot is moving on a straight line with a constant velocity $\mathbf{v}_A = \dot{\mathbf{a}}_i(t_0)$. The position of robot is simply calculated by

$$\mathbf{a}(t) = \mathbf{a}(t_0) + (t - t_0)\mathbf{v}_A, t \geq t_0, \tag{10.3}$$

where t_0 is the initial time and $a(t)$ is the position of robot at the current time t respectively. With respect to an approaching obstacle O, a set of relative ve-

locities $\mathbf{v}_{AO} = \mathbf{v}_A - \mathbf{v}_O$, possible to cause the collision within a short duration τ, are defined as a velocity obstacle,

$$\mathcal{VC}^{\tau}_{AO} = \{\mathbf{v}_{AO} | \forall t \in [0, \tau], \|\mathbf{a}_A - \mathbf{a}_O + t\mathbf{v}_{AO}\| \leq r_A + r_O\}. \tag{10.4}$$

Beside moving objects, an obstacle also includes other robots in local range to avoid mutual collisions among them. The velocity obstacle $\mathcal{VO}^{\tau}_{AO}$ is presented by a non-convex space within a truncated cone, limited by three straight lines. The collision is avoided if the velocity of robot lies outside the velocity obstacle $\mathbf{v}_A - \mathbf{v}_O \notin \mathcal{VO}^{\tau}_{AO}$. This is expressed by three linear constraints, $\mathbf{n}^l_{AO}.\mathbf{v}_{AO} \leq b^l_{AO}$, with $l \in 1, 2, 3$

$$\begin{aligned} \begin{bmatrix} \cos(\alpha + \beta) \\ \sin(\alpha + \beta) \end{bmatrix} \mathbf{v}_{AO} \leq 0, \\ \begin{bmatrix} \cos(\alpha - \beta) \\ \sin(\alpha - \beta) \end{bmatrix} \mathbf{v}_{AO} \leq 0, \\ -\frac{\mathbf{p}_{AO}}{p_{AO}}.\mathbf{v}_{AO} \leq \frac{p_{AO} - \bar{r}_{A+O}}{\tau}, \end{aligned} \tag{10.5}$$

where $\mathbf{p}_{AO} = \mathbf{a}_A - \mathbf{a}_O, \bar{r}_{A+O} = r_A + r_O, p_{AO} = \|\mathbf{p}_{AO}\|, \alpha = \arctan 2(-\mathbf{p}_{AO})$, and $\beta = \arccos(\bar{r}_{A+O}/p_{AO})$. The two first constraints are the two left and right half planes outside of the cone while the last one is to prevent the collision up to $\tilde{t} = \tau$. At least one of constraints has to be satisfied, therefore extra binary variables are introduced to choose the constraint to be applied,

$$\begin{aligned} \xi^l \mathbf{n}^l_{AO}.\mathbf{v}_{AO} \leq b^l_{AO} \\ \sum_l \xi^l \geq 1, \xi^l \in \{0, 1\}, l \in 1, 2, 3. \end{aligned} \tag{10.6}$$

10.2.3 Congestion control constraint

The obstacle avoidance constraint helps to push robots far away from the moving directions of obstacles in a local range. However, if there are many robots

routed into one place, a robot as trying to avoid each other may need a very long time to escape from the crowd. This may lead to a deadlock situation where a robot is not able to approach to its goal. A proposed solution to mitigate this issue is control congestion in such as way that a limited number of robots are allowed to enter a narrow crowded area and a robot should avoid a path with many intersections. To do so, each robot needs to be planned with several paths to the goal and to be able to choose the best among them to avoid congestion. The robots in the same working space should communicate and work together to find an optimal configuration which satisfies all.

As multiple paths are planned for each robot, the intersections of the paths construct a graph where the vertices represent the cross of path and the links connects two vertices on the moving paths of robots. A robot should avoid a node or a link which is potentially occupied by several robots at the same time. To establish alternative global paths, a random generation of paths like rapidly exploring random tree (RRT) [6] is one of the options. The path can also be found by a manual configurations from users. However, in this paper, in order to prioritise the travelling path with an optimal minimal distance, a conventional global path planning is executed multiple times. The paths found from the previous search are dilated and inserted into the binary map as static obstacles so that the next path is separately found. The alternative path plays an important role if the robot needs to pass through a narrow area with the possibility of facing another robot. On an empty space, the multiple paths created by this approach can be close to each other. Therefore, the separation is defined as the terminal condition to decide to keep searching the next path or not. Let a path be presented by a sequence of points in 2-D (dimensional) space. With two paths denoted as $S_1 = \{s_1^1, s_2^1, ..., s_{l_1}^1\}$ and $S_2 = \{s_1^2, s_2^2, ..., s_{l_2}^2\}$ where l_1, l_2 are the number of points of the two paths respectively. Let $d_1^1, d_2^1, ..., d_{l_1}^1$ be a set of minimum Euclidean distance of each point $\forall i, s_i^1$ to S_2 and $d_1^2, d_2^2, ..., d_{l_2}^2$ a set of minimum Euclidean distance of each point $\forall i, s_i^2$ to S_1. The separation of two paths is measured by $d(S_1, S_2) = \max\{d_1^1, d_2^1, ..., d_{l_1}^1, d_1^2, d_2^2, ..., d_{l_2}^2\}$. Also, at most N paths are allowed for each robot to reduce the complexity of

the whole system.

After multiple ways to the goal are established, the congestion control is realised by a constraint to limit the number of robots passing through a particular area and a utilisation term to penalise the paths with many crossings as shown in Figure 10.1. Obviously, there is a risk that two or more robots get stuck inside a narrow and long region, e.g., a corridor, when they approach each other in opposite directions. Therefore, the former term decides the bound on how many paths get access to the same link. In general, the congestion constraints for n-paths going through a narrow area ζ is given by

$$\mathcal{CC}_\zeta = \{\sum_{i=1}^{n} z_i^{k(i)} \leq c_\zeta | i \in \mathcal{A}, k \in \mathcal{V}_i\} \tag{10.7}$$

when robot i chooses the path $k(i)$ and c_ζ is the maximum number of robots allowed go through the conflict area ζ.

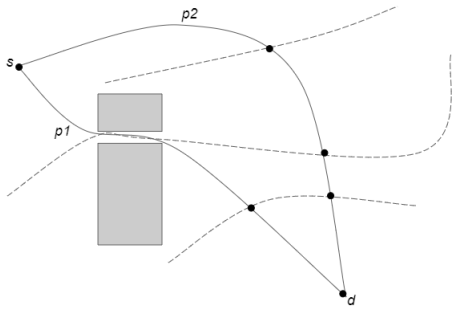


Figure 10.1: An example of possible congestion with multiple paths. The fist path p_1 goes through a narrow corridor with high changes of blocking another robot entering the corridor from two opposite sides. Meanwhile, the other path p_2 also faces some risks of collisions with three crossings.

10.2.4 Static obstacle avoidance constraint and motion constraint

The requirement of no collisions with static obstacles on the environment is stated by the static obstacle avoidance constraint as defined above in equation (10.2) as

$$\mathcal{SC}^{[t_i^0,t_i^1]} = \{a_i(t)|\mathcal{F}_i(a_i(t)) \cap \mathcal{S}_r = \emptyset, \qquad \forall t \in [t_i^0, t_i^1]\}. \tag{10.8}$$

For each robotic platforms, there are certain limits on the acceleration, steering angles, etc. dependent on the dynamic model of robots. Those limits are transformed into a set of motion constraints. The motion constraints accept only a set of valid velocities of forward trajectories $u(s_0, \mathbf{v}, t)$ respecting to all dynamic constraints from the initial state of robot $s_0 = [\mathbf{a}_i(t_0), \dot{\mathbf{a}}_i(t_0), \ddot{\mathbf{a}}_i(t_0), ...]$

$$\begin{aligned}\mathcal{MC}^{\tau} = \{\mathbf{v}|t \in [0,\tau], u(s_0, \mathbf{v}, t) \text{ satisfies to all} \\ \text{dynamic and kinematic constraints}\}.\end{aligned} \tag{10.9}$$

Instead of including $\mathcal{SC}^{[t_i^0,t_i^1]}$ and $\mathcal{MC}^{\tau}$ constraints in velocity space, a sampling method is applied to generate a set of valid trajectories within a window time in a configuration space, i.e. the space of feasible locations where a robot is able to reach. This mechanism of trajectory rollouts has been well developed in DWA-planner [18] in ROS.

10.2.5 Completed problem

The main objection function of the problem includes two components. The first quadratic term is to minimise the robot velocity close to the selected path and to make the moving trajectories smooth. Let $\mathcal{V}_i = \{k|k \in 1, 2, ..., p_i\}$ be a set of the available paths for robot i, p_i be the number of paths, $\mathcal{P}_i = \{\bar{\mathbf{v}}_i^1, \bar{\mathbf{v}}_i^2, ..., \bar{\mathbf{v}}_i^{p_i}\}$ be a set of preferred velocities on each path. The control velocity $\mathbf{v}_i$ to determine the next move of the robot is forced to be close to one of the preferred

velocities as only one path is chosen among $\mathcal{V}_i$. Let $\mathbf{z}_i = [z_i^1, z_i^2, ..., z_i^{p_i}]^T$ be the binary vector to select the path, $z_i^k \in \{0, 1\}$. The quadratic term $Q_i(\mathbf{v}_i, \mathbf{z}_i)$ is defined as follows,

$$\begin{aligned} Q_i(\mathbf{v}_i, \mathbf{z}_i) &= \|\mathbf{v}_i - \sum_{k=1}^{p_i} \bar{w}_i^k z_i^k \bar{\mathbf{v}}_i^k\|^2 + (\mathbf{v}_i - \mathbf{v}_i^{t-1})^T \mathbf{L} (\mathbf{v}_i - \mathbf{v}_i^{t-1}) \\ \text{s. t.} \quad & \sum_{k=1}^{p_i} z_i^k = 1 \end{aligned} \tag{10.10}$$

where $\bar{w}_i^k$ is the weight of each path, $\mathbf{L}$ is a semi-positive matrix to penalise the rapid changes in speeds and orientations [9]. The other non-quadratic term of the optimisation problem is given by

$$\begin{aligned} R_i(\mathbf{z}_i) &= \sum_{k=1}^{p_i} z_i^k \left(w_s \bar{s}(z_i^k) + w_c \bar{s}(z_i^k \right) \\ \text{s. t.} \quad & \sum_{k=1}^{p_i} z_i^k = 1 \end{aligned} \tag{10.11}$$

where w_s and w_c are the weights of two terms, $\bar{s}(z_i^k) = \int_{t_i^0}^{t_i^1} a_i(t)\, dt$ is the total length of travelling path from time t_i^0 to t_i^0, and $a_i(t)$ is the position of robot i at time t. The second term related to the number of crossing on the path and is computed by $\bar{c}(z_i^k) = \sum_l \mathcal{S}_l$ where robot i chooses the path k and there is an intersection of the path k with the intersection l. Here $\{S_l | l = 1, ..., L\}$ is a set of the weight of intersections, S_l is computed by the number of paths passing intersection l, and L is the total number of intersections.

The joint optimisation function for all robots to find their optimal paths and

velocities is expressed by,

$$\begin{aligned} F(\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n, \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n) &= \sum_{i=1}^{n} Q_i(\mathbf{v}_i, \mathbf{z}_i) + R_i(\mathbf{z}_i) \\ \text{s. t.} \quad \sum_{k=1}^{p_i} z_i^k &= 1, \forall i \in [1, n]. \end{aligned} \tag{10.12}$$

Along with the cost function, a set of constraints are defined to find collision-free paths for robots considering that they have different options to chose their paths and also need to avoid any dynamic obstacles on their moving ways.

Finally, the overall optimisation problem is formulated, in which the optimal control velocities and selected global paths $[\mathbf{v}^*_{1:n}, \mathbf{z}^*_{1:n}]$ are estimated in a joint manner,

$$\begin{aligned} [\mathbf{v}^*_{1:n}, \mathbf{z}^*_{1:n}] = &\underset{[\mathbf{v}_{1:n}, \mathbf{z}_{1:n}]}{\operatorname{argmin}} F(\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n, \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n) \\ \text{s. t.} \quad &\sum_{k=1}^{p_i} z_i^k = 1, \forall i \in [1, n] \\ &\mathcal{CC} \quad \text{(Constraint 1)} \\ \mathbf{v}_i, \mathbf{v}_j \notin \mathcal{VO}^{\tau}_{ij}, \forall i, j \in \mathcal{A} \quad &\text{(Constraint 2)} \\ \mathcal{SC}_i \ \ \forall i \in \mathcal{A} \quad &\text{(Constraint 3)} \\ \mathcal{MC}^{\tau}_i \ \ \forall i \in \mathcal{A} \quad &\text{(Constraint 4)} \end{aligned} \tag{10.13}$$

10.3 Decentralised Optimisation with OSQP

Basically, it requires highly computational resources to solve equation (10.13) in a central manner with respect to the number of discrete variables proportional to the number of robots in the working space. Another drawback is that all robots have to depend on a single node to receive their assigned path before they start moving. Therefore, this paper focuses on a decentralised im-

plementation to tackle the optimisation where each robot estimates its optimal solution separately. The joined optimisation problem is divided into two parts: global path selection (Section 10.3.1) and local movement planning (Section 10.3.2). The global path planning focuses on finding a set of optimal paths $\mathbf{z}^*_{1:n}$ among multiple choices with the objective $G(\mathbf{z}_{1:n}) = \sum_{k=1}^{n} Q_i(\mathbf{v}^c_i, \mathbf{z}_i) + R_i(z_i)$ and Constraint (1), where $\mathbf{v}^c_{1:n}$ is the current velocity vector of each robot i. Once the global path is chosen, the optimal velocity is found by optimising $\sum_{k=1}^{n} Q_i(\mathbf{v}_i, \mathbf{z}^*_i)$ with fixed term $\mathbf{z}^*_{1:n}$. Instead of solving $\sum_{k=1}^{n} Q_i(\mathbf{v}_i, \mathbf{z}^*_i)$ in a centralised manner to find the optimal set $\mathbf{v}^*_{1:n}$, the utility $Q_i(\mathbf{v}_i, \mathbf{z}^*_i)$ with Constraint (2) is optimised locally on each robot to find v^*_i. The rest of variable $v_j, \forall j \neq i$ in $Q_i(\mathbf{v}_i, \mathbf{z}^*_i)$ is fixed from the last known velocity $\mathbf{v}^c_{1:n}$. Finally, to satisfy Constraint (3) and Constraint (4), the optimised path and velocity is mapped from a velocity space into a configuration space in Section 10.3.3 to address the collisions with static obstacles and to bound the motions of robots within their dynamics limits.

10.3.1 Optimal global path search

Once a robot receives the multiple planned paths from the others, it performs the search of the optimal global path to start moving. However, the complexity of solving $G(\mathbf{z}_{1:n})$ with congestion constraints increases rapidly with respect to the number of robots. Let m be the maximum number of paths assigned for each robot, a worst-case complexity of the problem is $O(m^n)$, which is exponential to the number of involved robots n. When n is small, a flat search with implicit enumeration is deployed. All combinations of binary variables are generated. Infeasible variables are pruned and the remaining is examined to find the best solution. As the proposed solution aims at a light and fast algorithm for mobile robot, the approximate search is applied when n is large. Similar to the approaches presented in [19] and [20], not all but only a subgroup of robots should be included in the search. The paths of other robots are fixed with the last-known information. Maximum κ searching iterations is performed with the limits of n' robots. Therefore, the complexity is reduced

to $O(\kappa m^{n'})$. Once a robot finish a single search, it broadcasts its optimal selection, which is utilised by other robots to formulate the next search. The search stops when the maximum number of iterations is reached or the optimal paths broadcasted by other robots remain unchanged in consecutive updates. The overall algorithm to find an optimal path for each robot is summarised in Table 6.

Algorithm 6 Global optimal path search.

Collect multiple paths assigned for each robot $\mathcal{V}_i, \forall i = 1, ..., n$
$k \leftarrow 0$
$\Omega_i \leftarrow$ a set of n' robots to be included in the optimal search for robot i
while $k < \kappa$ **do**
  Collect optimised path selection from all robots
  Generate the all combinations of $z_j^{k(j)}, \forall j \in \Omega_i$, $k(j)$ is the number of paths of robot j
  Prune infeasible combinations to generate a pool for the search
  Find the optimal path of robot i with optimised $G(\mathbf{z}_{1:n})$ from the pool
  Broadcast the optimised path found for robot i
end while

10.3.2 Mixed integer quadratic optimisation with OSQP

The optimisation of the utility $Q_i(\mathbf{v}_i, \mathbf{z}_i^*)$ with Constraint (2) for a robot i is described in Section 10.2.2 in the form of the MIQP with binary variables to select linear constraints. In general, an MIQP is defined as

$$\begin{aligned} &\text{minimise } f(x) = \frac{1}{2} x^T \mathbf{L} x + c^T x \\ &\text{subject to } l \leq \mathbf{A} x \leq u \\ &\qquad x_i \in \mathbb{Z}, \forall i \in \mathbb{I}, \end{aligned} \tag{10.14}$$

where $x \in \mathbb{R}^n$ is a variable vector, $\mathbf{L}$ a $n \times n$ symmetric positive semidefinite matrix, and a vector $c \in \mathbb{R}^n$. The linear constraints are given by the $m \times n$

matrix A with the lower bound $l \in \mathbb{R}^m$ and upper bound $u \in \mathbb{R}^m$. Here, m is the number of linear constraints and n is the number of variables respectively. Integer variables belong to the set $\mathbb{I}$ with the total number $|\mathbb{I}|$. The MIQP is $\mathcal{NP}$-hard in general where the computation grow exponentially if the simple approach of exhaustive search is used to iterate all integer combinations. Numerous approaches to find optimal solution for MIQP are described in [21]. Among them, *branch-and-bound* is one the most efficient approach when considering about practical and commercial aspect. In this method, the search is performed over a tree to explore feasible regions of integer variables and solving the quadratic problem (QP) in the form,

$$\begin{aligned} &\text{minimise } \frac{1}{2}x^T\mathbf{L}x + c^T x \\ &\text{subject to } l \leq \mathbf{A}x \leq u \\ &\qquad\qquad \underline{x_i} \leq x_i \leq \bar{x_i} \in \mathbb{Z}, \forall i \in \mathbb{I}. \end{aligned} \tag{10.15}$$

The algorithm starts with the root node, QP$(-\infty, \infty)$, to find the first initial solution $\bar{x}$. Once the problem QP$(\underline{x}, \bar{x})$ is solved with the fractional solution, the branch creates two left (L) and right (R) nodes, QP$(\underline{x}^L, \bar{x}^L)$ and QP$(\underline{x}^R, \bar{x}^R)$, with the parent bound $(\underline{x}, \bar{x})$ from one of non-integer elements $\tilde{x_i}$, $(\underline{x}^L, \bar{x}^L) = (\underline{x}_i, \lfloor \tilde{x_i} \rfloor)$ and $(\underline{x}^R, \bar{x}^R) = (\lceil \tilde{x_i} \rceil, \bar{x}_i)$. The heap $\mathcal{H}$ is used to store the leaves in buffer before exploration. The important property of the *branch-and-bound* allows sub-branches to be pruned to reduce searching space. The algorithm is described in Algorithm 7 as follows.

To solve the sub-problem QP$(\underline{x}, \bar{x})$ efficiently, the alternating direction method of multipliers (ADMM) [22] with OSQP solver is used (Algorithm 8). The OSQP solver is well developed for embedded optimisation as it utilises the heuristics with ADMM to be able to find solutions with good timing results. Algorithm 2 describes OSQP in details where $\rho, \sigma > 0$ are the step size parameters, α is the relaxation parameter, and Π is the Euclidean protection operator.

With regards to the limitation of powers on embedded device, numerous

Algorithm 7 Branch-and-bound algorithm for MIQP.

$U \leftarrow \infty, \mathcal{H} \leftarrow QP(-\infty, \infty)$
while $\mathcal{H} \neq \emptyset$ **do**
  pop QP($\underline{x}, \bar{x}$) from $\mathcal{H}$
  solve QP($\underline{x}, \bar{x}$) to get $\tilde{x}, f(\tilde{x})$
  if QP($\underline{x}, \bar{x}$) is infeasible $\vee$ $f(\tilde{x}) > U$ **then**
    prune the current node
  else if $\tilde{x}$ is integer feasible **then**
    $U \leftarrow f(\tilde{x}), x^* \leftarrow \tilde{x}$
    prune nodes in $\mathcal{H}$ with their lower bound $> U$
  else
    choose a vector $\hat{x}$ where its elements $\hat{x}_i$ are integer
    from the solution $\tilde{x}, i \in \mathbb{I}$
    if $\hat{x}$ is infeasible $\wedge$ $f(\hat{x}) < U$ **then**
      $U \leftarrow f(\hat{x}), x^* \leftarrow \hat{x}$
      prune nodes in $\mathcal{H}$ with their lower bound $> U$
    end if
    branch node QP($\underline{x}, \bar{x}$)
  end if
end while

Algorithm 8 QSQP algorithm.

$x^0 \leftarrow \rho, y^0 \leftarrow \alpha, y^0 \leftarrow \sigma$
while termination condition does not meet **do**
  solve $[\tilde{x}^{k+1}, t^{k+1}]^T$ from

$$\begin{bmatrix} L + \sigma I & A^T \\ A & -\frac{1}{\rho} I \end{bmatrix} \begin{bmatrix} \tilde{x}^{k+1} \\ t^{k+1} \end{bmatrix} = \begin{bmatrix} \sigma x^k - q \\ z^k - \frac{1}{\rho} y^k \end{bmatrix}$$

  $\tilde{z}^{k+1} \leftarrow z^k + (1/\rho)(t^{k+1} - y^k)$
  $x^{k+1} \leftarrow \alpha \tilde{x}^k + (1-\alpha) x^k$
  $z^{k+1} \leftarrow \Pi\left(\alpha \tilde{z}^{k+1} + (1-\alpha) z^k + (1/\rho) y^k\right)$
  $y^{k+1} \leftarrow y^k + \left(\alpha \tilde{z}^{k+1} + (1-\alpha) z^k - \rho z^{k+1}\right)$
end while

heuristics and searching strategies have been developed to be able to return sub-optimal solution within time constraints. There are three main directions including tree exploration, selection of branching variables, and computation of integer solution. The proper combination of those heuristics for embedding environment has been explained in [21].

10.3.3 Movement planning in configuration space with motion constraints and static obstacle avoidance

In order to include static obstacles into the search in VO-space, each static obstacle is by a VO with a polygon footprint and zero velocity. Consequently, the number of both continues and discrete variables involved in the MIQP increases proportionally to the number of static obstacles detected by the robot. If the dynamics constraints of the robot are also considered, the complexity of the MIQP becomes high to be handled effectively in the embedded hardware of mobile robots. The solution for this issue is found by integrating the found optimal global path z_i^* and velocity v_i^* into the DWA-planner available in ROS. According to the motion constraints configured based on the robot's model, the forward trajectories of robots are sampled in configuration space. Trajectories which possess collisions with static obstacles are discarded. The remained trajectories are evaluated with DWA-cost functions to choose the best one moving toward the goal and following the global path. Three additional implementations are introduced. First, DWA is configured with the optimal selection z_i^* among multiple global paths to the goal. Second, a trajectory is also removed if the velocity of a robot on the trajectory (v_i^ω where ω is the index of the trajectory) violates Constraint (2), meaning that a robot has a risk to be collided with others in a VO-space. The velocity v_i^ω is calculated by dividing the different vector between the end and start point of the trajectory with the travelling duration of the robot on the trajectory ω. Third, the new cost function to minimise $\|v_i^\omega - v_i^*\|$ is added into DWA. Apparently, all constraints are considered at the final stage with DWA to find the best optimal movement for each robot,

satisfied in both velocity space and configuration space.

10.4 Experimental Results

The proposed approach is implemented on the open source ROS [23] with the Kinetic Kame version. The robot model Turtlebot 3 Burger [24] is used in all experiments. This robot has a dimension of $0.14m \times 0.18m \times 0.19m$ (L$\times$W$\times$ H), maximum translational velocity $0.22m/s$, and maximum rotational velocity $2.84rad/s$. The robot is also equipped with 360^0 laser distance sensor and 9-DOF IMU for localisation and obstacle detection.

The simulations are executed on an Ubutu 16.04 Linux computer with a quad-core 2.9Hz Intel i7 processor and 16GB memory. The comprehensive Gazebo 9 simulator [25] is used. The dynamic motion and shape properties of a simulated robot are configured with the same values from a real robot. The navigation tasks are evaluated on a complex maze scenario [26] with a square size $14m \times 14m$. A total of 8 robots are used in this experiment where their stating positions are roughly placed on a circle. The goals are uniformly distributed on the map where the minimum distance between the start and goal is $2m$ and the separation between two goals is also $2m$. The visualisation of the terrain configurations and robots placing on their starting locations in Gazebo is give in Figure 10.2.

For congestion control, two restricted areas are defined to allows the maximum of 2 robots passing through. Also each robot is assigned maximum two global paths to its goal. Global paths and those restricted area are visualised by Rviz ROS [27] in Figure 10.3(A). Correspondingly, an example of an intersection map generated for congestion control is given in Figure 10.3(B).

The experiments are repeated 20 times, correspondent to 160 navigation tasks assigned to 8 robots. The results of running conventional DWA are compared with the proposed decentralised path planning (DPP) using only obstacle avoidance (OA) constraint and using both OA and congestion control (CC). With regards to CC constraints, the global optimal path search (Algorithm 6)

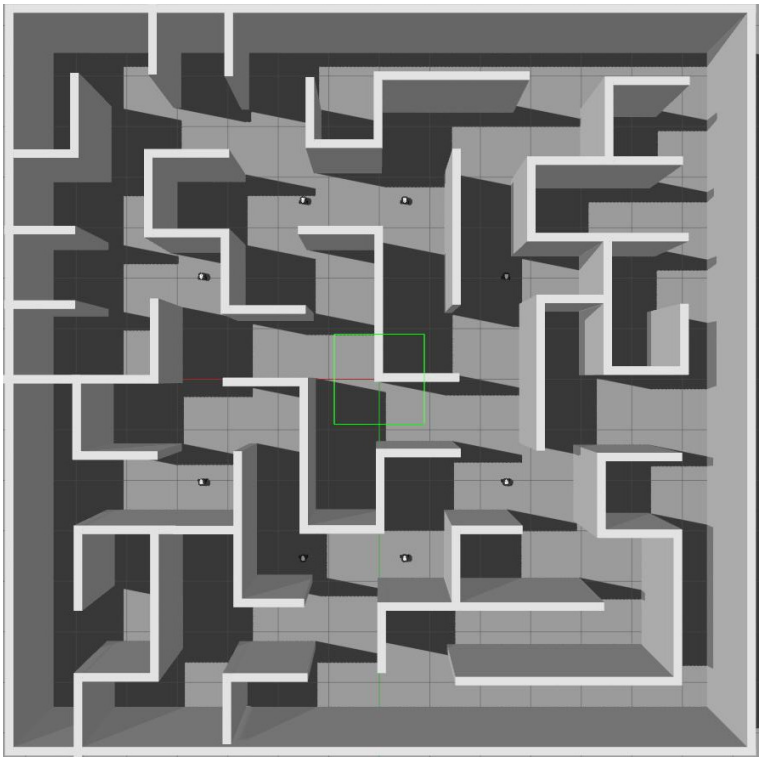
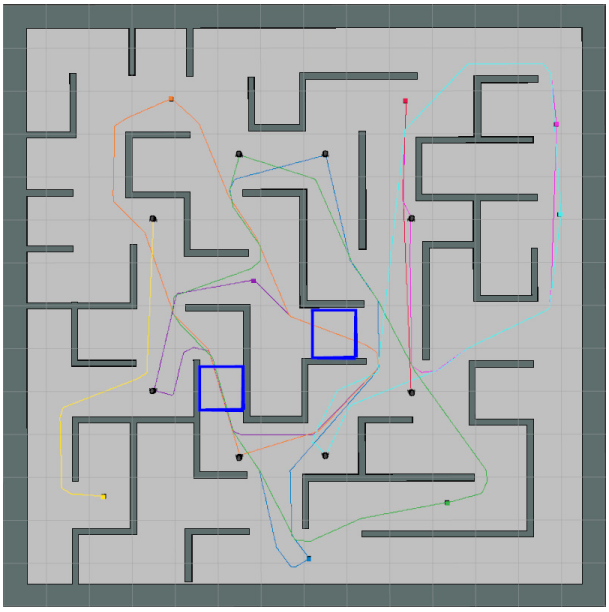
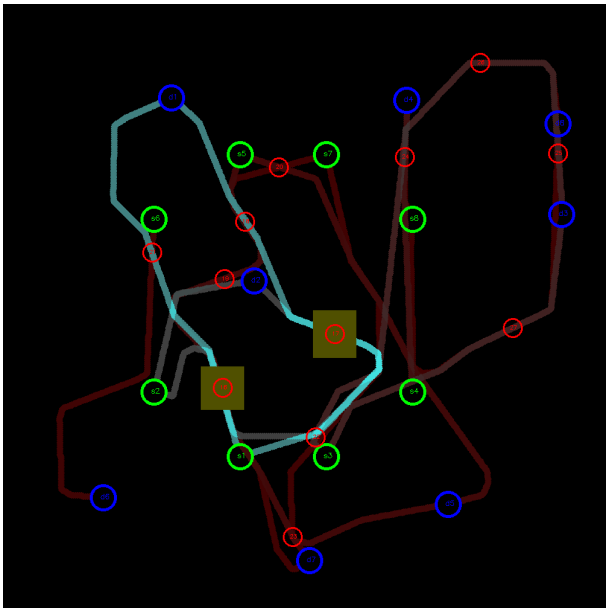


Figure 10.2: The maze terrain with 8 Turblebot 3 Burger in Gazebo simulator.



(A)



(B)

Figure 10.3: (A) Multiple global paths and (B) intersection map.

needs to be performed at every robot. For a full search, the robot seeks for the optimal global within the maximum 256 combinations of two paths for each robot. For the limited search presented in this experiment, a robot considers maximum 16 combinations within a group of 4 robots and the search is repeated twice.

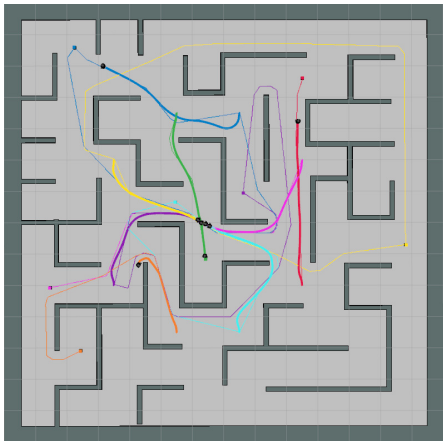
When only DWA is used, head-on collision happens if robots do not take into account the moving directions to avoid each other. Once OA constraint is applied, the head-on collision is mitigated but robots are facing more deadlocks with traffic jams in narrow areas. The utilisation of CC helps route a part of robots away from a heavy traffic area, correspondingly reducing dead-locks. The overall results presented in Table 10.1 reveal that the proposed DPP with OA+CC has reduced the deadlocks significantly even with a limited search and that the OA constraint has played an important role to avoid head-on collisions.

Table 10.1: Comparisons between DWA with the proposed DPP.

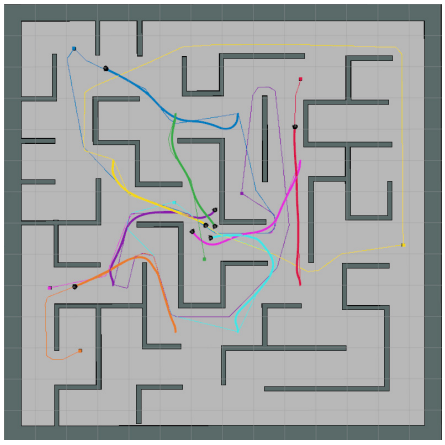
	DWA	DPP w. OA	DPP w. OA+CC *(limited search)*	DPP w. OA+CC *(full search)*
Head-on collision	37/160	2/160	4/160	0/160
Deadlock	65/160	44/160	34/160	28/160

10.5 Conclusion

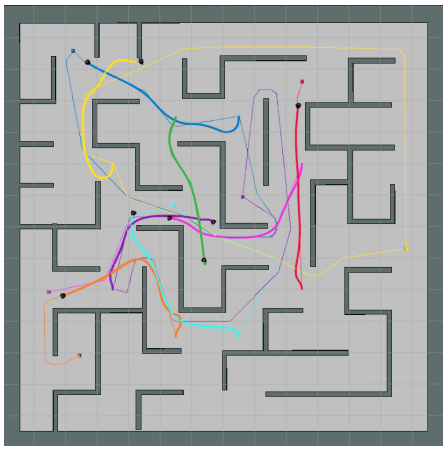
This paper has introduced a decentralised approach to tackle the navigation tasks for multiple robots. The algorithm is divided into three stages. The first stage deals with global path selection. Each robot has alternative paths to approach to its goal, therefore it needs to choose the path which has not only a minimal travelling distance but also less interference with path from other robots. Continuously, an approximate velocity amplitude and direction has to be found for each robot to be aligned with the optimal path found from the



(A)



(B)



(C)

Figure 10.4: (A) DWA, (B) DPP with OA, and (C) DPP with OA+CC.

previous stage and to avoid the collisions with other moving robots. Finally, the assigned moving velocity is verified in a configuration space to ensure that the movement of robots satisfy the dynamic configuration of robots and no collisions between the robot and static obstacles from the environment. The optimisation is performed separately on each robot where the robots have to broadcast their positions, moving velocities, multi-global paths, and optimal path selection. The whole framework has been evaluated with experiments to solve the effectiveness of combining congestion and obstacle avoidance into path planning to reduce deadlocks and increase the successful rate of finish the navigation tasks. In addition, the optimisation with MIQP problem is solved with OSQP, which is an effective solver suitable with the limited computational resource equipped on mobile robots.

In future works, a faster optimisation with machine learning can be deployed to reduce the computational load on the mobile robots. To be applicable in a large scale, proposed approach can be designed in a hierarchy manner where robots close together in physical space are grouped into zone. The moving within a zone is performed as normal. In other cases, robots are routed into a gateway at the boundary of the zone to transfer from one zone into another. This approach actually adapts the divide-and-conquer mechanism into the path planning problem. Finally, a large-scale evaluation will be planned with hundreds of robots in experiments.

Acknowledgement

The research leading to the presented results has been undertaken within the research profile DPAC - Dependable Platform for Autonomous Systems and Control project, funded by the Swedish Knowledge Foundation.

Bibliography

[1] G.-Z. Yang et al. , "The grand challenges of Science Robotics," *Science Robotics* , vol. 3, no. 14, eaar7650, 2018.

[2] S. Russell and P. Norvig, "Artificial intelligence a modern approach," 4th ed., Boston, Pearson, 2018.

[3] W. Zeng and R. L. Church, "Finding shortest paths on real road networks: the case for A*," *International Journal of Geographical Information Science*, vol. 23, no. 4, pp. 531–543, 2009.

[4] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the International Conference on Robotics and Automation*, pp. 3310–3317, 1994.

[5] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[6] LaValle, M. Steven, J. Kuffner Jr., and J. James, "Randomized kinodynamic planning," *The International Journal of Robotics Research (IJRR)*, vol. no. 5, pp. 378–400, 2001.

[7] A. Koubaa, "Robot Operating System (ROS): The complete reference," First Volume, Springer, 2016.

[8] P. Fiorini and Z. Shiller Z, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760-772, 1998.

[9] J. Alonso-Mora, A. Breitenmoser, M. Rufli, PA. Beardsley, and R. Siegwart R, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," *Distributed Autonomous Robotic Systems Springer Tracts in Advanced Robotics*, vol. 83, pp. 203-216, 2010.

[10] J. Snape, J. v.d. Berg, J. P. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacles," *IEEE Transactions on Robotics*, vol. 27, pp. 696-706, 2011.

[11] D. Wilkie, J. v.d. Berg, and D. Manocha, "Generalized velocity obstacles," in *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[12] M. de Weerdt and B. Clement, "Introduction to planning in multiagent systems," *Multiagent and Grid Systems*, vol. 5, no. 4, pp. 345–355, 2009.

[13] G. Sharon, R. Stern, A. Felner, and N.R. Sturtevant, "Conflict based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 291, pp. 40-66, 2015.

[14] D. Silver, "Cooperative pathfinding," in *Proceeding of the Fist AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pp. 117-122, 2005.

[15] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proceeding of the AAAI19-Thirty-Third AAAI conference on Artificial Intelligence*, pp. 7643-7650, 2019.

[16] J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics*, vol. 32, no. 5, 1163-1177, 2016.

[17] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, 637-672, 2020.

[18] A. Filotheou, E. Tsardoulias, A. Dimitriou, A. Symeonidis, and L. Petrou, "Quantitative and qualitative evaluation of ROS-enabled local and global planners in 2D static environments", *Journal of Intelligent and Robotic Systems*, vo. 98, pp. 567–601, 2019.

[19] B. A. Asfora, J. Banfi and M. Campbell, "Mixed-integer linear programming models for multi-robot non-adversarial search," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6805-6812, 2020.

[20] G. Hollinger, S. Singh, J. Djugash, and A. Kehagias, "Efficient multi-robot search for a moving target," *The International Journal of Robotics Research*, vol. 28, no. 2, pp. 201-219, 2009.

[21] B. Stellato, V. V. Naik, A. Bemporad, P. Goulart, and S. Boyd, "Embedded mixed-integer quadratic optimization using the OSQP solver," in *Proceedings of the European Control Conference (ECC)*, 2008.

[22] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1-122, 2011.

[23] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, Leibs, Jeremy, Wheeler, Rob, and A. Y. Ng, "ROS: An open-source robot operating system," presented at the *ICRA Workshop on Open Source Software*, 2009.

[24] Turtlebot 3 [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/.

[25] Gazebo [Online]. Available: http://gazebosim.org/.

[26] Z. Yan, L. Fabresse, J. Laval, and N. Bouraqadi, "Metrics for performance benchmarking of multi-robot exploration," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[27] Rviz ROS [Online]. Available: https://github.com/ros-visualization/rviz.