PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Towards Reliable Robotics: from Navigation to Coordination
Auteur: Author:	Majda Moussa
Date:	2019
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Moussa, M. (2019). Towards Reliable Robotics: from Navigation to Coordination [Ph.D. thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/4153/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/4153/
Directeurs de recherche: Advisors:	Giovanni Beltrame
Programme: Program:	Génie informatique

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Towards Reliable Robotics: from Navigation to Coordination

MAJDA MOUSSA
Département de génie informatique et génie logiciel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie informatique

Décembre 2019

© Majda Moussa, 2019.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Towards Reliable Robotics: from Navigation to Coordination

présentée par **Majda MOUSSA**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Foutse KHOMH, président
Giovanni BELTRAME, membre et directeur de recherche
Christopher J. PAL, membre
Philippe GIGUÈRE, membre externe

DEDICATION

Sometimes we need a little magic...
Aymoun, Linix, Mum, Dad
you are my magic

To the memory of all those who
we have loved and lost

ACKNOWLEDGEMENTS

I would like to thank everybody I had the chance to meet and work with during my doctoral studies. Particularly, I would like to express my sincere gratitude to my advisor Giovanni Beltrame for his insightful advices, persistent support and help.

I also acknowledge all my committee members : Professor Foutse Khomh, Professor Chris Pal, and Professor Philippe Giguère for generously offering their time to review and evaluate this document.

I am grateful for my husband Aymen for his constant love, care and support. I would like to thank him for holding me up during my bad time and for his deep confidence in my abilities and competencies.

I would like also to thank my parents, my mother-in-law and siblings whom, despite the distance, helped me to overcome the challenges through moral and emotional support.

A special thought goes to my colleagues at MistLab with whom I shared the office for years, I appreciate their friendship and I wish them all the best.

RÉSUMÉ

Les **robots autonomes** et les **systèmes multi-robots** ont connu un intérêt sans cesse croissant par les scientifiques et l'industrie. Plusieurs applications telles que les robots assistants, les robots gestionnaires de stock ainsi que les véhicules autonomes nécessitent des algorithmes de **navigation** et de **coordination** fiables pour permettre leur déploiement dans des environnements dynamiques et relativement méconnus. Ainsi, la **capacité d'adaptation** est une caractéristique fondamentale permettant une utilisation accrue et une intégration plus facile des systèmes multi-robots. Afin de posséder cette agilité d'adaptation, les robots devraient opter vers un comportement assez robuste avec une aptitude à réajuster leurs actions selon la cinématique de l'environnement. Ce mémoire de thèse, s'interesse aux problèmes de fiabilité lors du déploiement des systèmes multi-robots dans des environnements dynamiques et inconnus. Il s'articule autour de deux contributions majeures, à savoir : Un mécanisme de planification et de réajustement de mouvement quasi optimal qui roule à une fréquence allant jusqu'à 200 Hz. Ainsi qu'un framework de vérification de la **robustesse** des comportements coopératifs des systèmes multi-robots.

La première contribution a été inspirée de l'habilité de quelques animaux à naviguer en se fiant au champ magnétique terrestre. En effet, nous avons constaté que le champ magnétique n'admet pas de maxima locaux, ce qui permet aux animaux de suivre son gradient. Par conséquent, un robot est capable de parcourir tout type d'environnements en faisant propager un champ magnétique virtuel et en suivant son gradient. Toutefois, la résolution des équations de Maxwell, qui décrivent la physique des champs magnétiques, est complexe et nécessitent des simulations numériques couteuses en termes de ressources et temps de calcul. Pour pallier cette difficulté, nous proposons un approximateur de la solution des équations de Maxwell basé sur un réseau de neurones profond entrainé exclusivement sur des solutions provenant de simulations numériques avancées. L'environnement est représenté par une carte de conductivité. Nous affectons une conductivité maximale à la destination du robot et une conductivité nulle aux obstacles. Le calcul de la distribution du champ magnétique virtuel permettra au robot de suivre le gradient qui le mènera vers sa destination selon un chemin quasi optimal.

La solution proposée a des performances supérieures aux algorithmes de navigation les plus sophistiqués. Elle assure toujours la complétude et l'optimalité du chemin prédit en temps réel (200 Hz). Nous avons mis à l'épreuve l'efficacité de notre méthode par des simulations basées sur la physique d'un véhicule aérien sans pilote et un vehicule utilitaire équipés tous les deux d'un LIDAR. Nous avons également mené des essais sur un vrai vehicule de course de petite échelle. Par ailleurs, on met en évidence la possibilité d'utiliser notre approche en jeu vidéo et pour effectuer des vols en

troupeau.

La capacité d'un robot à réajuster ses déplacements à une grande fréquence est cruciale pour une navigation fiable vers une destination spécifique dans des environnements dynamiques. Cela est également très important pour permettre à un système multi-robot de déterminer ce que la destination devrait être pour chaque robot-membre. Dans l'absence d'un planificateur centralisé (ce qui est fort probable si nous voulons éviter un point de défaillance unique), les systèmes multi-robots distribués utilisent souvent la communication et le consensus pour assigner des objectifs et des tâches aux robots-membres.

Dans notre deuxième contribution, nous étudions la capacité des robots à établir des comportements consensuels fiables (tels que la répartition des tâches ou l'élection du chef) dans un environnement instable, où la communication est sujette aux erreurs et les robots peuvent être défectueux. C'est une première étape pour relever les défis qui pourraient se poser lors de la conception d'une navigation multi-robot efficace. Nous utilisons les méthodes formelles et en particulier la technique de Model Checking statistique pour fournir des garanties probabilistes de la robustesse d'un ensemble de comportements bien connus. Nous proposons un framework formel qui permet de modéliser les *propriétés temporelles* des stratégies consensuelles en tenant compte de l'incertitude relative aux problèmes de connectivité et à la défaillance des robots. Nous avons utilisé des propriétés quantitatives pour exprimer les exigences du système. Nous avons mis en évidence l'efficacité de notre framework en simulation et avec des vrais robots, cela à différents ordres de grandeur. Il était intéressant de constater que nous étions en mesure de vérifier la dynamique du système tout en faisant abstraction de la mobilité des robots. Le framework proposé est évolutif et peut assurer une vérification statistique rapide (en quelques minutes) aux comportements consensuels pour des systèmes composés de milliers de robots.

À notre avis, les travaux présentés dans cette thèse préparent le terrain au déploiement effectif de comportements multi-robots dans des environnements dynamiques, ainsi qu'une nouvelle perception des systèmes robotiques de point de vue biologique et physique.

ABSTRACT

Autonomous robots and **multi-robot systems** are of growing interest for industry and academia. Many real-world applications such as assistive robotics, inventory management, and autonomous driving require reliable **navigation** and **coordination** algorithms that can be deployed in a partially unknown, dynamic environment. The ability to **adapt** is a key feature for the widespread use and societal integration of multi-robot systems. To achieve this adaptation ability, robots must implement inherently robust behaviors and must be sufficiently fast to re-plan their actions when their environment changes. This dissertation deals with the problem of reliably deploying a group of robots in a dynamic, unknown environment, and provides two key contributions: a mechanism for robots to plan and re-plan their motion near optimally up to 200 times per second; and a framework to verify the **robustness** of multi-robot cooperative behaviors.

For the first contribution, observing how some animals are able to navigate using the Earth's magnetic field, we realize that this is possible because the magnetic field has no local maxima, and animals can follow its gradient. This means that a robot can navigate any kind of environment by propagating a known virtual magnetic field and following its gradient. However, solving Maxwell's equations–which govern the physics of magnetic fields– is complex and demands computationally costly numerical simulations. To overcome this problem, we propose a deep neural network as an approximator for Maxwell's equations, exclusively trained on high-quality numerical simulations. We model the environment as a conductivity map with its maximum in a goal location and zero for obstacles. After computing the virtual field propagation, a robot can follow the virtual magnetic gradient to optimally reach the goal.

Our solution outperforms the state-of-the-art navigation algorithms by ensuring completeness and providing an optimal path in real-time (up to 200 times per second). We demonstrate the effectiveness of our method with physics-based simulation of a LIDAR-equipped unmanned aerial vehicle and a sport utility vehicle, as well as real-world experiments using a racing rover. Furthermore, we show how the approach can be applied to flocking and gaming.

This ability to re-plan at high speed is crucial for one single robot to reliably navigate towards a specific goal in dynamic environments, but it is also crucial to make a group of robots able to determine what the goal should be for each individual robot. In the absence of a centralized planner (a likely occurrence if we want to avoid a single point of failure), distributed multi-robot systems most often use communication and consensus building to assign goals and tasks to individual robots.

In our second contribution, we study how robots can reliably build consensus-based behaviors (such as task allocation or leader election) in an unreliable world, where communication is error-

prone and robots are subject to failures. This is a first step to tackle the challenges that might be encountered during the design of an effective multi-robot navigation system. We use formal methods and particularly statistical model checking techniques to provide probabilistic robustness guarantees for a set of widely-used behaviors. We propose a formal framework based on a model of the *timing properties* of consensus strategies under uncertainty, *i.e.* connectivity issues and the fallibility of robots, that uses well-defined, measurable properties to express system requirements. We perform extensive testing of our framework, taking into account failure probabilities at different orders of magnitude, both in simulation and with real robots. Interestingly, we show how we can verify multi-robot system dynamics all while abstracting the mobility of robots. Our system is scalable and can provide statistical model checking of consensus properties for behaviors with thousands of robots in a matter of minutes.

In our opinion, the work presented in this thesis paves the way for the realistic deployment of multi-robot behaviors in dynamic environments, as well as a fresh look at robotic systems from a biological and physical perspective.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

AI	Artificial Intelligence
APF	Artificial Potential Field
CL-RRT	Closed-loop-Rapidly-Exploring Random Trees
CNN	Convolutional Neural Network
ERRT	Extended-Rapidly-Exploring Random Trees
GBP	Graph-Based Planners
GPS	Global Positioning System
GA	Genetic Algorithms
INS	Inertial Navigation System
MAS	Multi-agent System
MC	Model Checking
NN	Neural Network
NPTA	Network of Priced-Timed Automata
PSO	Particle Swarm Optimization
PTA	Priced-Timed Automata
RRT	Rapidly-Exploring Random Trees
RT-RRT*	Real-Time-Rapidly-Exploring Random Trees Star
SBP	Sampling-Based Planners
SLAM	Simultaneous Localization and Mapping
SMC	Statistical Model Checking
SUV	Sport Utility Vehicle
UAV	Unmanned Aerial Vehicle
VS	Virtual Stigmergy
UVs	Unmanned Vehicles

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

This dissertation covers the research work pursued to fulfil the requirements of the Philosophiae Doctorate degree in Computer Engineering within the MIST laboratory of Polytechnique Montréal (Québec, Canada) from September 2014 to November 2019. This document covers 2 submitted articles presented as separate chapters. This chapter introduces the topics we explored, gives an overview of the problems we tackled and enumerates the research contributions we achieved during our research journey.

1.1 Research Context and Motivation

Autonomous robots have raised wide interest in recent years by leveraging the most advanced technologies in software, artificial intelligence and machine learning. Their use has evolved from deploying a single robot for deliveries [1] or inspection [2] to employing large numbers of cooperative robots for autonomous surveillance [3] and environment monitoring [4]. Fig. 1.1 shows some uses of a variety of mobile robotic systems at different scales. All those applications require mobile robots that are capable of navigating autonomously unknown environments, detecting and avoiding obstacles in real-time.

When it comes to dealing with dynamic surroundings, adaptability –the ability of a robot to deal with unforseen circumstances– becomes an essential requirement to ensure reliable and robust operation. That is, robots should be sufficiently reactive to perceive the surrounding world, deftly act on what they see, and re-plan their actions accordingly.

The ability to adapt at high-speed level is of paramount importance for distributed multi-robot systems as well. A simple example would be a group of robots autonomously agreeing on the allocation of a set of goals using communication. Being adaptable to changing goals and assignments allows synchronized behaviors and efficient cooperation.

In this research project, we focus on the problem of reliably deploying mobile robotic systems –whether that be a single robot or a multi-robot system– in a dynamic and partially unknown environment. Fig. 1.2 offers an overview of the topics of interest and their logical relationship. We shed light on the problem of real-time embedded navigation and multi-robot coordination robustness. This research is split in two work packages (WPs), namely adaptive navigation and robust coordination. These two contributions are key features for implementing inherently reliable behaviors for multi-robot systems *e.g.,* cooperative exploration and pattern formation.

The navigation process of an autonomous vehicle can be defined as the combination of three fun-

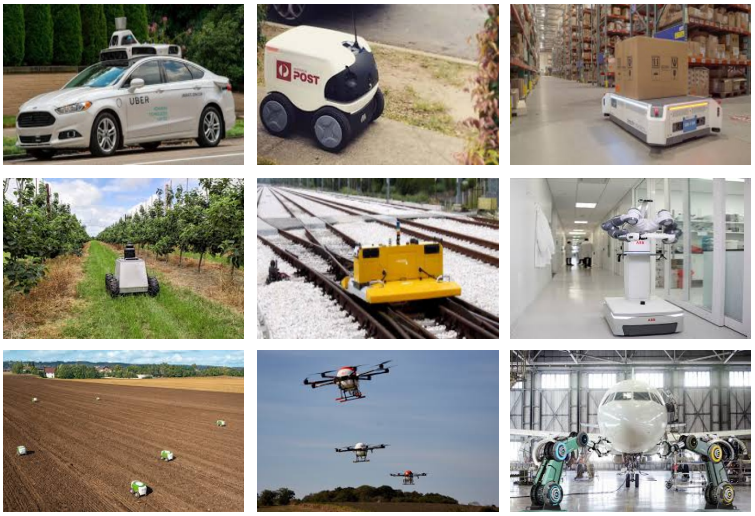


Figure 1.1 Mobile robots applications.

damental sub-processes; i) perception, ii) self-localisation, and iii) motion planning. A perception sub-process allows a robot to collect information about its surroundings using a variety of sensors. A motion planning sub-process uses the collected information conjointly with a localization sub-process to build a local map and generate a feasible path from a starting point to a goal position. Given the generated path, a motion controller is then called to drive the robot motion and steer it to the desired goal.

A well-established navigation process should address the aforementioned common designs to ensure that the robot i) reacts effectively to unpredictable situations within a reasonable time; ii) considers multiple concurrent requirements in the process; and iii) reaches its target with a good compromise between time and distance. However, several challenges have to be taken into consideration namely i) the inherent uncertainties in a partially unknown and unstructured environment; ii) incomplete perceptual information and, iii) inaccurate actuators and sensors.

Another side of this problem revolves around cooperative navigation, which is particularly interesting to explain certain collective behaviors observed in biology such as in schools of fish [5], flocks of birds [6], and ant colonies [7, 8].

Through natural selection, animals evolved navigation-specific attributes to suit their environments and their own physical and sensory characteristics. For instance, various species such as birds have developed a magnetic sensor able to detect the intensity and the direction of Earth's magnetic field. This information, along with other perceptual and shared cognitive knowledge, allow flocks of birds to efficiently travel South and back during their migratory seasons. They act collectively as tiny brains linked together through various forms of communication. Their ability to map the

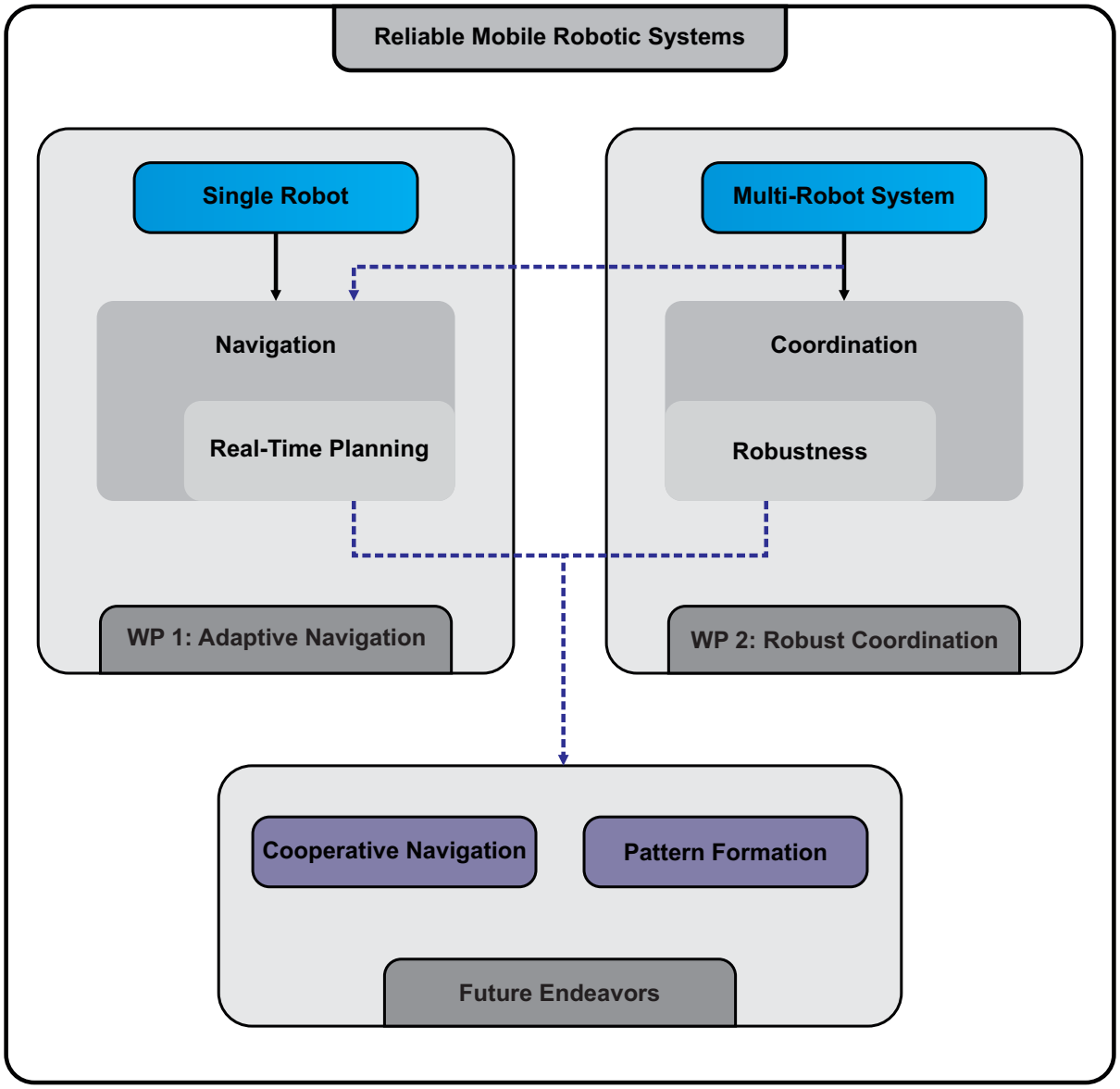


Figure 1.2 Overview of the main work packages in this research project.

environment as a magnetic distribution, helps them to navigate around the globe each year without getting lost [9].

A common feature of these diverse biological systems it that they maintain a certain coordination among the group members even though decisions are taken at the individual level in a fully distributed way. That is, a collective navigation behavior is the result of propagating shared information between the individuals that behave as a network. A good example of this behavior is observed in the cooperative transport mechanism of longhorn crazy ants, *Paratrechina longicornis* [10]. These ants are highly efficient at jointly navigating maze-like obstacles while maintaining coordination even when the nest direction is blocked.

The study of these multi-agent models is crucial for understanding many complex phenomena related to animal behavior, and designing distributed navigation and control strategies for large scale multi-agent systems referred to as swarms [11, 12].

As the robots that typically make up a swarm have only local perception and very limited local communication abilities, one of the challenges in designing swarm navigation processes is to understand the effect of the communication process on the group performance. Swarms work together through simple local interactions with their neighbors or their environment. The quality of swarm behaviors that emerge from these interactions (e.g. decision making, exploration, collective motion, shape formation, mapping) typically depends on the robustness of the communication process [13]. The latter should consider i) connectivity issues, ii) environment uncertainty and failures, iii) sensor interference and severely limited communication bandwidth.

Since biological systems are able to fulfil the navigational needs of animals, they can be considered as an important source of inspiration for developing creative robotics navigation solutions. In this dissertation, we demonstrate the efficiency of a bio-inspired navigation system developed to boost the ability of a mobile robotic system to navigate in a dynamic and partially unknown environment. The system is designed to improve the performance and overcome the limitations of existing navigation algorithms. This is achieved by combining localization, map building and path planning with three intelligent competencies namely learning, reasoning, and optimizing. The proposed system takes into consideration the distributed property of multi-agent systems and swarm-robotic systems as well as the level of responsiveness required in such real-time systems. To overcome these challenges, we studied the communication process of groups of robots collaborating together to perform a specific task by sharing information.

That being said, whereas research on AI-driven robots is making meaningful contributions today, further exploration of this ever-evolving field can lead to better artificial systems able to assist humans in more complicated tasks. In this dissertation, we make use of physical laws and AI to foster the application of robotics in the real world and pave the way for robot scientists to advance

the discipline.

1.2 Problem Statement

This dissertation presents research that provides the community with a real-time, distributed navigation algorithm. The solution is easily deployed on a single robot as well as a swarm of robots to perform collaborative tasks. We study the problem of autonomous planning and navigation in a partially unknown environment with the objective of reaching the goal in minimum time and distance. Overall, our research intends to provide a solution able to mitigate multiple challenges, including, but not limited to:

— The difficulty to respect real-time requirements in a highly-dynamic space whereby robots should avoid moving and static obstacles when they come into view;

— The lack of a way to engineer bio-inspired systems to produce intelligent navigation solutions;

— The fully distributed aspect of swarm systems implies that robots make decisions based on their own reasoning and local interactions. This requires a robust communication system. Therefore, we should deal with:

  – The lack of a formal definition of what a robust and resilient communication is and what robustness entails;
  – The lack of formal models and methodologies to assess or quantify robustness;

— The inefficiencies induced by the limited computing capabilities of mobile robots whether they operate solely or in teams–the navigation solution should be computationally efficient–.

Tackling these challenges has the potential to substantially advance the application of robotics in the real world.

1.3 Research Objectives

We outline a set of research objectives in the following:

1. Propose a responsive and scalable bio-inspired navigation system using virtual magnetic fields (Chapter 3):

   (a) Implement a deep learning framework to solve Maxwell's equations;

(b) Build a path optimization module based on the AI prediction of the environment magnetic field distribution;

(c) Implement the approach on a real hardware *i.e.* an autonomous rover;

2. Address the need for a formal robustness analysis of swarms coordination processes with the aim of implementing swarm navigation with probabilistic reliability guarantees (Chapter 4):

(a) Provide a sound methodology to enhance the present understanding of swarm-based interactions. In that, we account for communication issues and robot failures to assess and quantify the swarm level of robustness;

(b) Model the swarm system using probabilistic timed automata to be able to verify the robustness of swarm behaviors using Statistical Model Checking (SMC);

(c) Conceive a solution to enhance the scalability of the proposed SMC-based model– taking into consideration systems of thousands of robots–;

(d) Provide a proof-of-concept that demonstrates how Objective 1 can be applied in the context of swarm robotic system (Chapter 3).

1.4 Contributions and Impact

To the extent of our knowledge, the contributions (Work Packages, WPs) presented in the body of this dissertation and outlined here are original in nature and live up to advance both academic and technological knowledge:

— WP 1: A new approach models the path planning problem for mobile robots in a dynamic environment. The approach leverages natural magnetic field laws to endow robots with a virtual magnetic sense. An optimal and stable trajectory can be then planned from any starting point to a specific goal position. This research work is entitled "Real-Time navigation with Virtual Magnetic Fields" by Majda Moussa and Giovanni Beltrame. It has been submitted to Science Robotics, 2019. Current status: Under Review;

— WP 2: A formal model to analyze consensus protocols for robot swarms. In particular, we leveraged a statistical model-checking approach to verify the consensus-based strategies for a networked robotic swarm in terms of the communication quality and robot failure probability. The work enhances the present understanding of the consensus-based behaviors, thereby determining their level of robustness. This research work is entitled "On the robustness of consensus-based behaviors" by Majda Moussa and Giovanni Beltrame. It is submitted to Swarm Intelligence, 2019. Current Status: Under Review, 3rd revision.

The scientific significance of the proposed work and its impact on both academia and industry include:

— An original and robust AI-based navigation solution that joins academic research and the needs of industrial robotics applications *e.g.,* real-time embedded navigation solutions.

— a sound strategy to certify and guarantee the correctness of robotic swarms consensus-based behaviors, for the purposes of i) guiding swarms design decisions and verifying that safety constraints have been met and, ii) analyzing and assessing the swarm communication process in an attempt to better understand swarm interactions and their modalities as an important feature of cooperative navigation;

Given the growing interest in mobile intelligent robots, a real-time distributed navigation solution is a *sine qua non* for the advancement of many of tomorrow's robotics missions. While nature is often the inspiration behind the most ingenious systems, it may play a leading role in reshaping the way robotic systems are conceived. Literature on robotics and automation has pointed to the impressive potential of the diversity of natural laws and processes in creating artificial intelligent system. However, more effort is needed to deeply mine this potential and use it to drive change in the robotic field which places the work presented here at the forefront to fulfill this need.

Focused on swarm intelligence field, this research has as well, the potential of improving the robustness and the resiliency of robotic systems behaviors that require a formal notion of consensus. In that this research set useful stepping stones for the creation of a new generation of fault-tolerant behaviors that can be sustainably executed for longer periods of time, supporting more resilient and robust swarm robotic system.

1.5 Thesis Outline

The dissertation follows the traditional structure of a thesis by article which dictates that the body of the contributed articles, published and submitted, be presented in separate chapters.

— Chapter 1 familiarizes the reader with the topics addressed throughout this dissertation. It prefaces the research problem and lists the research objectives, contributions and the impact of the work;

— Chapter 2 places the work presented among the current literature;

— Chapter 3 presents a first part of the contribution body of this dissertation: An optimal real time navigation algorithm for mobile robots (WP 1);

— Chapter 4 presents a second part of the contribution body of this dissertation: A statistical formal framework to assess the robustness of swarms consensus based behaviors (WP 2);

— Chapter 5 discusses the contributions of the previous chapters and highlights the relations among them;

— Chapter 6 wraps the dissertation with concluding remarks, and future avenues;

— The document concludes with a number of appendices that gives a glimpse at the works achieved during my PhD and have not been presented here.

CHAPTER 2 LITERATURE REVIEW

This chapter examines the fundamental understanding and existing research that addresses mobile-robot intelligence. We consider three types of mobile-robot systems namely single robot, multi-robot systems and swarm robotic systems. The difference between a multi-robot system and a swarm robotic system lies mainly in their sizes and cooperative process. That is to say, swarms generally consist in a larger number of simple robots (generally homogenous) which coordinate in fully distributed manner to perform a collective behavior, whereas a multi-robot system consists generally on smaller number of (possibly) heterogeneous powerful robots which may achieve a global task without any communication between each other referring to centralized commands. A navigation process that can be equally deployed on those different systems should account for several considerations and issues concerning uncertainties, imprecision and incomplete information in an unknown and hazardous environment. In the following, we extend on the current navigation approaches to highlight the problems of existing mobile navigation algorithms taking into consideration the three introduced systems.

2.1 Mobile Robot Navigation Problems

In applications of advanced robotics, path planning is definitely a challenging problem especially for systems that operate in a highly dynamic environment. This process employs a set of sensors for perception and cognition and provides a set of control commands that will be executed by the robot's actuators. In the last decade, a massive number of real-time path planning algorithms have been proposed providing solutions for different scenarios. The efficiency of those algorithms has been widely studied and discussed [14, 15]. In this section, we review the relevant literature regarding the currently used approaches in real-time path planning. The most common approaches to path planning fall into the categories of Graph-Based Planners (GBP) [16], Sampling-Based Planners (SBP) [17], Artificial Potential Field (APF) [18] and heuristic methods [19].

Graph-based search methods stem from graph theory. The environment is represented by a grid where the robot can only move along the grid edges (routes between adjacent grid cells) and can take position only on discrete locations (grid nodes). In this category, many algorithms have been proposed to plan paths between a start node and a goal node including the breadth first search or grassfire algorithm [20], Dijkstra's algorithm and the A* method (approaches implementing those algorithms are discussed in [16]). A major drawback of these approaches is they become computationally expensive with larger space environments especially if a fine-grain resolution is required for the task. Moreover, these approaches take an additional stage after exploring the environment

and building the graph to extract the path from the graph. This can be more challenging if we deal with multi-query tasks, where the task is intended to find a path to different goal points.

More advanced solutions address the previous problems by using a sampling approach. Rapidly-Exploring Random Trees (RRT) [21] employ randomness to quickly exploring a large search space with iterative refinement. However, it cannot guarantee finding the optimal path. Most recently, variations of RRT (such as RRT* [22], RRT*-smart [23]) have been proposed. They eventually converge to a better solution than RRT. Nevertheless, they usually suffer from large number of nodes and require additional post-processing algorithms to enhance the path smoothness which slows down the convergence rate and increase computational time. Moreover, these approaches do not provide any theoretical guarantees for reaching an optimal solution. Instead, they ensure a weaker notion of completeness referred to as probabilistic completeness; that is to say, finding a path is guaranteed (if one exists) given sufficient runtime of the algorithm which has been shown to be infinite for some cases.

The real-time versions of RRT-based approaches, such as RRT^X [24], Extended-Rapidly-Exploring Random Trees (ERRT) [25], Closed-loop-Rapidly-Exploring Random Trees (CL-RRT) [26] and Real-Time-Rapidly-Exploring Random Trees Star (RT-RRT*) [27], have implemented new strategies to limit the search space and reduce computing time by interleaving space exploration with taking actions. However, they still inherit the same limitations of their predecessors. The most recent approach RT-RRT* claims to be the best candidate to RRT-based real-time planning as it provides shorter path and are able to handle multi-query tasks. RRT-based approaches require an iterative processing to smooth the path. These approaches are also sensitive to implementation parameters (*i.e.* step size, goal bias, number of iterations, etc.) which must be carefully chosen according to the applications needs. In contrast, we aim at providing a solution that does not require any tuning effort and able to predict an optimal path in a real-time fashion.

The Artificial Potential Field (APF) method [18] consider the environment as a distribution of a potential field: the goal generates an attractive potential whereas the obstacles produce a repulsive one. This allows the robots to navigate to their goal while avoiding the obstacles. Many research works such as [28–30] have implemented APF method to perform navigation. For instance, Szulczyński *et al.* [28] proposed a new obstacle avoidance technique based on APF whereby they used harmonic functions *i.e.* Laplace equation. The approach takes into account elliptical obstacle described in two-dimensional environment with static and dynamic goal. In [31], the authors have made use of electrostatics to produce the potential function and determine collision-free paths in real-time. Nevertheless, APF-based approaches come with an inherent known problem *i.e.* local minima. To overcome this challenge, this method has been blended with a variety of heuristic and AI algorithms such as Genetic Algorithms (GAs) [32], pseudo bacterial GAs [33], fuzzy logic [34],

Particle Swarm Optimization (PSO) [35] and cell decomposition [36] that use the potential as a cost function. Although these approaches have reduced the limitations of APF, they have added an overhead which made the real-time requirement difficult to fulfill.

More recent works such as in [37, 38] resort to neural networks to perform planning using GBP with minimum loss in performance. They used GBP as "training expert" to generate feasible paths. Although these approaches enable fixed-time path generation, they have the same limitations as the algorithms they try to imitate. More particularly, the work presented in [37] deals solely with static environments. That is, for every new environment a new dataset should be collected and the model should be retrained. Other neural planners such as [39] have targeted self-driving cars to provide optimal trajectories by interpreting intermediate representations of the environments. The neural model takes as input the raw data of the environment and outputs a cost value for each position that the self-driving car can take within the planning horizon.

2.2 Single-Robot Versus Swarms Navigation

A robotic swarm is a self-organizing multi-robot system that involves a high number of robots to perform a variety of collaborative navigation tasks [40] such as formation keeping [41] and flocking [42]. The system does not depend on any external infrastructure and is agnostic to any form of centralized control. Being relevant to engineering applications, swarm robotics have been widely investigated by the scientific community and are deemed to promote designing and creating fault tolerant, scalable and flexible systems.

The design of a scalable robot swarm system relies on the miniaturization of its single robots members. That is, the capabilities of a single robot are limited and simplified to make the swarm less demanding of resources and more efficient in terms of cost and productivity. This requires the system to operate based on a cooperative approach for the purpose of conducting meaningful behaviors that are difficult to be fulfilled separately at the individual level. However, the challenges that resolve around those limitations are very hard to overcome, and are compelling for robust but simple navigation strategies that depend upon simple communication protocols.

Cooperation and coordination are key factors in the design of a swarm navigation system [41, 43, 44]. This process is insured through local interactions and sensing [45]. That is, a failed communication generally results in increasing conflicts, imprecision and uncertainty leading to non-efficient and non-robust navigational system. These problems may become more and more severe in dynamic and unknown surroundings whereby the swarm is required to satisfy real-time constraints considering for any possible collision with other robots and static obstacles.

Navigation strategies that are conceived for single mobile robots usually rely on navigation maps

[46], map-building strategies [47, 48], and external infrastructure [49]. They are apparently inadequate for swarms given the resources limitations associated with their simple and independent physical robots. Swarm intelligence research have found inspiration in some biological systems [50] namely social creatures such as insects and animals that live in groups. Several works [51–53] have promoted the characteristics of those biological systems in the realization of autonomous, distributed, and self-organizing artificial swarm robotics systems.

In biology, the entire process of a swarm behavior is insured by individual local interactions– which allow collective coordination based on shared information and sensing. Being inspired by this natural process, a plethora of research works have recognized the need to i) address the existing challenges of the collective interaction design– including the satisfaction of the real-time constraint, the dynamic and continuous change of the operating environment, and the imprecision of the actuators and sensors, ii) study all the system's parameters to enhance the robustness of the communication during the development of the navigation system.

2.3 Swarm Robotic Coordination

The consensus-based decision making process is one of the most efficient coordination and control mechanisms used for distributed multi-agent systems. The cooperative transport behavior that can be observed among groups of ants in nature requires in fact, that all the ants in the group individually adjust their own understanding of the travel direction in a way to converge to a common belief. Broadly, this mechanism appears in nature in various forms namely swarming of honeybees [54], flocking of birds [55], migration of animals [56], and has inspired many engineering applications such as coordination of decentralized block-chains [57], synchronization of coupled oscillators [58], and formation of autonomous vehicles [59]

Stigmergic interactions [50, 60] are an universal consensus-based coordination mechanism that allows to share data and ensure convergence among agents within a dynamic environment. The idea is that independent actors leave signs or emit signals in the environment. Those flags will, in turn, be sensed by other actors and help to determine and incite subsequent actions. Biological multi-agent systems use, in fact, stigmergic collaborations to exchange information. They modify their environment, which triggers a future response. For instance, ants use pheromones, wasps use secretions and people use wikis. Therefore, a better understanding of stigmergy and groups dynamics offers new insights into the world of multi-agent coordination, which is the essence of swarm intelligence.

In swarm robotics, behaviors requiring consensus have attracted great attention over the years. They have been used in a wide range of applications that require robustness, resilience and flexibility [43,

61, 62]. A collective decision is particularly prone to deadlocks–fail to form consensus [63] which can result in a split decision, or no decision at all. A plethora of approaches and methodologies have addressed this issue and have proposed fault-tolerant behaviors such as task allocation [64], formation control [65], leader election [66], and determination of coordination-based variables, e.g., rendezvous time [67].

2.3.1 Coordination Modeling and Verification

Swarm robotic systems have been widely applied to numerous critical applications such as disaster rescue, autonomous surveillance and environment monitoring [68]. Therefore, they need to sustainably operate for extended periods of time without human intervention and display a high degree of tolerance to faults. Inherent scalability and robustness are often considered as essential properties of a swarm system as it typically relies on decentralized control and local communication. To maintain a convenient level of robustness for robot-swarm behaviors, it is thus essential to develop efficient approaches and methodologies to actively check and accommodate faults.

Simulations and real-world experiments have been usually used to assess the correctness of a variety of consensus-based behaviors. For instance, Pinciroli *et al.* [66] have proposed a new use of the stigmergy concept referred to as virtual stigmergy. The applicability of the proposed concept is demonstrated using simulations and experiments. As we argue in Chapter 4, those methods have been proven to be inefficient and unreliable as they are not capable to predict all the scenarios a swarm may encounter while operating in complex environments. Therefore, formal-model-based approaches [69] are more appropriate since they mathematically model all the possible behaviors of the system and assess their requirements using logic formulas.

Research works such as [70–73] have proposed a variety of formal-based methodologies to formally assess the correctness of some collective behaviors *e.g.,* foraging [70], swarm aggregation [72], live human navigation [74], *etc.*. Most of those methods have focused on modeling the *timing properties* of the behaviors of interest without accounting for, neither the quantitative aspect of the system nor the *uncertainty* which is one of the key features of a swarm robotic system. A notable exception is the work of Kouvaros *et al.* [75] which investigates a protocol allowing the swarm to reach consensus over a certain opinion. The authors propose a formal framework to quantitatively evaluate the fault-tolerance of the protocol of interest. Although the approach is promising, it does not give insights on the time to reach consensus. Furthermore, it only considers temporal epistemic properties which account only for certain knowledge acquired by agents during their interactions. No attempt was made to model and verify *uncertainty*.

CHAPTER 3 ARTICLE 1 - REAL-TIME NAVIGATION USING VIRTUAL MAGNETIC FIELDS

Preface

Inspired by biological sensing of the terrestrial magnetic field, we model path planning as a physics-based problem: the core idea of this work is to use the properties of the magnetic field to perform navigation tasks in autonomous robots. In particular, we develop a data-driven approach to rapidly and accurately solve Maxwell's equations so that we can compute the propagation of a *virtual* magnetic field in real-time in any kind of environment. This virtual field can be followed as a real-time path planning strategy for autonomous systems. Interestingly, our approach can be trained exclusively in simulation, and does not need any further adaptation to work on different platforms.

Authors Majda Moussa and Giovanni Beltrame
Submitted to: Science Robotics

Abstract

Humans and animals have learned or evolved to use magnetic fields for navigation. Knowing how these fields propagate can be used for motion planning. However, computing the propagation of electromagnetic fields in a given environment requires solving complex differential equations with advanced numerical methods, and therefore it is not suitable for real-time decision making. In this paper, we present a real-time approximator for Maxwell's equations based on deep neural networks that predicts the distribution of a virtual magnetic field. We show how our approximator can be used to perform autonomous 2D navigation tasks, outperforming state-of-the-art navigation algorithms, ensuring completeness, and providing a near-optimal path up to 200 times per second without any post processing stage. We demonstrate the effectiveness of our method with physics-based simulations of an unmanned aerial vehicle, an autonomous car, as well as real-world experiments using a small off-road autonomous racing vehicle. Furthermore, we show how the approach can be applied to multi-robot systems, video game technology, and can be easily extended to 3D problems.

3.1 Introduction

Autonomous vehicles have sparked a wide interest in recent years, and pushed the many advances in software, artificial intelligence and machine learning. Applications such as delivery, inspection, and monitoring require intelligent robots that are capable of navigating autonomously to their destinations, detecting and avoiding obstacles in real-time [76]. This type of path planning is a challenging problem, especially for systems that operate in a highly dynamic environment. Several start-ups (e.g. Embodied Intelligence and Realtime Robotics) have resorted to dedicated hardware solutions to solve the path planning problem in real-time, with a market estimated in the billions of dollars [77]. Although the path planning problem has some practically demonstrated solutions such as Waymo self-driving cars [78] and the Skydio drone [79], these implementations are not necessarily optimal, they require precise external positioning (e.g. GPS), high computational performance, and have a high energy cost.

In the last decade, a massive number of real-time path planning algorithms have been proposed, studied, and discussed [14, 15]. The most common approaches to path planning fall into the categories of graph-based methods [16], sampling-based planners [17], artificial potential fields [18], and heuristic methods [19].

In graph methods, the environment is represented by a grid where the robot can only move between adjacent grid cells and can take position only in discrete locations (grid nodes). Standard solutions in this category include Dijkstra's algorithm and A* [16]. A major drawback of graph methods is that they become computationally expensive with larger environments and higher resolutions. To address this problem, many solutions use a sampling approach: originally, Rapidly-Exploring Random Trees (RRT) [21] exploits randomness to quickly explore a large search space with iterative refinement. Variants of RRT added probabilistic guarantees that a path would be found if it existed – e.g. RRT* [22], RRT*-smart [23] – as well as real-time performance – RRT^X [24], ERRT [25], CL-RRT [26], and RT-RRT* [27]. However, these solutions are far from optimal: guarantees are only asymptotic (i.e. assuming we have infinite computation time) and real-time solutions trade off accuracy by interleaving planning and acting on partial plans. More recent works [37, 38] resort to neural networks to generate paths using graph-based planners as "training experts". Although these approaches enable fixed-time path generation, they have the same limitations as the algorithms they try to imitate.

To solve this issue, we found inspiration from the natural world. Evolution has addressed the problem using existing features of our planet: animals have long since learned to take advantage of Earth's magnetic field to navigate their environment. Various species, ranging from birds and mammals to reptiles and insects, are able to sense the magnetic field and, along with sunlight, use

it for navigation [80]. Wu and Dickman [9] show that some neurons in animal brains encode information on the magnetic field's direction, intensity and polarity. This reveals that the animals have a magnetic sense that provides them with an internal global positioning system and a magnetic compass for directional heading. This natural skill allows animals to navigate thousands of kilometers towards their destination without getting lost.

In essence, knowing how the magnetic field propagates, humans and animals alike can use the field as a map. Looking at large scales, the Earth's magnetic field is well modeled and studied [81–83], and currently used for satellite navigation [84, 85]. This is possible because the magnetic field does not have local maxima [86, 87] and its gradient can be used for global navigation.

While we can use an existing magnetic field (e.g. Earth's) for navigation, we surmise that a robot could navigate any kind of environment by propagating a *virtual magnetic field* and following its gradient. The lack of global maxima makes the gradient path *optimal* and *guarantees completeness* –that is, if a path exists, it will be found. This operation requires solving Maxwell's equations to determine the value of the virtual magnetic field across the environment. However, solving Maxwell's equations is complex and computationally expensive, and usually demands numerical simulations such as finite elements or finite differences methods [88]. The computational resources needed to run these simulations in real-time far exceeds the capabilities of a mobile robot or autonomous vehicle. Hussein and Elnagar [89] attempted the use of a magnetic potential field for path planning using the finite differences method. Their work showed promise, but they were not able to achieve real-time performance.

In this paper, we present a data-driven solver of Maxwell's equations that can compute the distribution of the magnetic field in any 2D environment up to 200 frames per second, making our solution suitable for path-planning in highly-dynamic and time-varying environments. We use an auto-encoder convolutional neural network exclusively trained in simulation that can predicts the magnetic field distribution in unknown environments with high accuracy. We showcase how this method is readily applicable to many navigation tasks such as outdoor/indoor path planning and gaming. We also show how our method is easily extendable to 3D environments.

3.2 Results

Our main result is a learning-based path planning system that provides optimal or almost optimal results for 2D environments. Interestingly, our path planner is not trained on existing path planners or other kinds of oracles, but rather on the well-known physics of the electromagnetic field.

Electrical conductivity, the ability of a medium to carry electrical currents, is strongly related to the magnetic distribution in a given environment [90]. Intuitively, one can describe an environment

as a conductivity map to compute the propagation of the magnetic field. However, the relation between the environment's conductivity and the propagation of the magnetic field is often indirect and strongly non-linear [91]. Convolutional neural networks, CNNs [92], are known to be very powerful approximators for arbitrary non-linear functions [93]. Often used in computer vision, CNNs work by learning patterns and making decisions at the level of pixels, based on local spatial information. A vast range of computer vision problems, the likes of image segmentation [94–96] and pixelwise prediction [97], have been solved using CNNs.

We extend this idea and create a deep learning framework that approximates the relationship between the conductivity of an environment and its corresponding magnetic field distribution. Given that the behavior of the magnetic field is well understood and easily simulated, we build an (arbitrarily) large dataset of conductivity maps and their corresponding field distribution using a commercial simulator. We train a CNN on this dataset and obtain an approximator of the magnetic field distribution. We use the magnetic field to generate an optimal path to any set point in the environment. The overall framework of our method is shown in Fig. 3.1.

The generation of conductivity maps is a key feature: we use 2D images where each pixel represents the conductivity value of a corresponding discrete area of the environment (of arbitrary size, chosen according to the desired accuracy). Studying the simplest practical form of this problem, we encode *obstacles* (anything that cannot be traversed) with zero conductivity ($\sigma_o = 0$), while we assign the highest conductivity to a *goal* location (σ_g). For the rest of the environment, we assign conductivity values to each pixel according to its distance to the goal. For every pixel $p(x, y)$, the conductivity $\sigma_e(x, y)$ is defined as:

$$\sigma_e(x, y) = \frac{c}{\sqrt{(x - x_g)^2 + (y - y_g)^2}} \tag{3.1}$$

where x_g and y_g are the goal coordinates and c is a fixed parameter referred to as the environment conductivity constant.

We transform the conductivity map into a triangular mesh (the geometrical model and mesh in Fig. 3.1A) and compute the magnetic field distribution via finite elements [88].

We train our neural network, called MaxConvNet, as a supervised pixelwise regressor using the conductivity data as input and the magnetic distributions as their corresponding ground truth (see Fig. 3.1B). Pixelwise regression is similar to the well-known pixelwise classification problem [98], however, each pixel in an image is assigned to a continuous value.

A well-trained neural network learns the general relation between input and output data and can accurately predict never-seen-before inputs. We tested MaxConvNet on new, randomly generated environments and compared the prediction with the ground truth magnetic field distribution per pixel. Fig. 3.2A depicts the loss curves of both training and testing phases: the testing loss curve

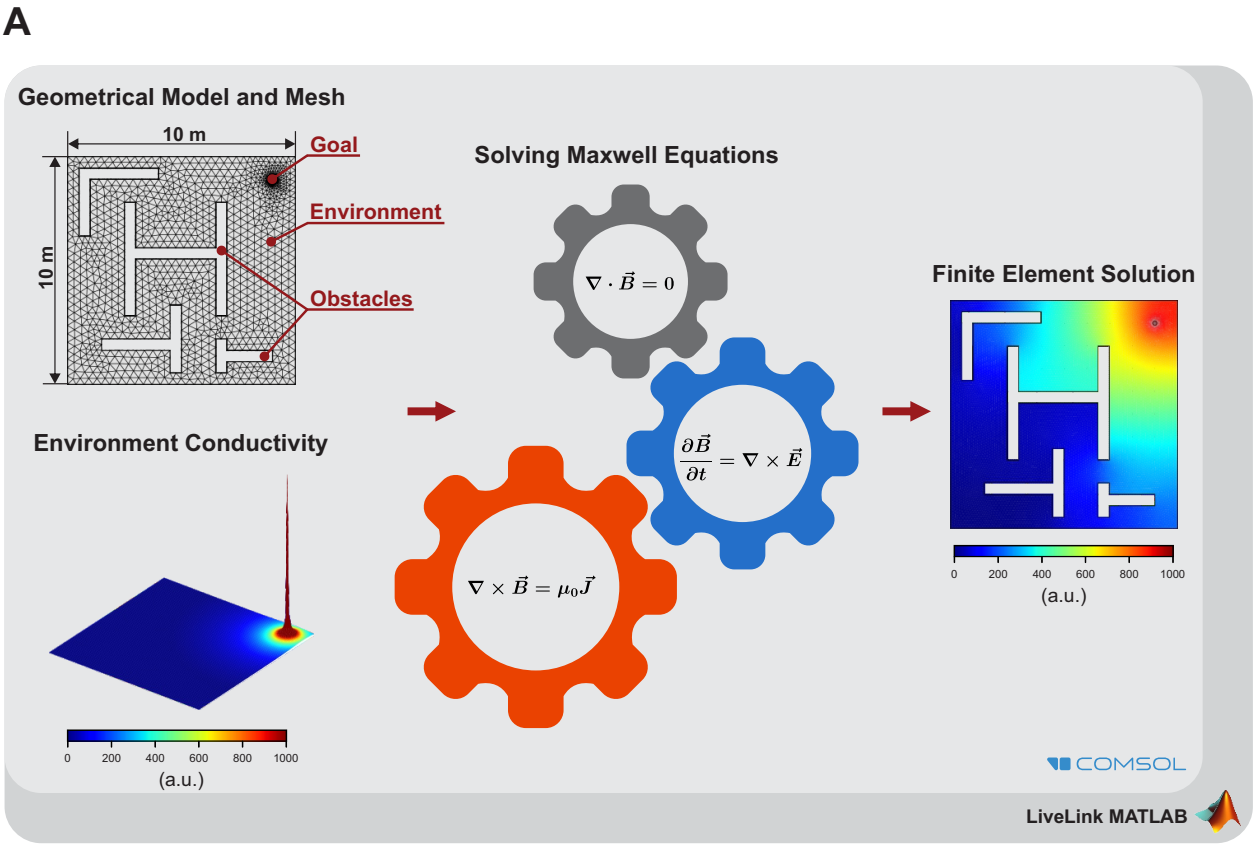


Figure 3.1 Deep learning framework for solving Maxwell's equations: (A) Schematic representation of the data collection process (conductivity data maps and the corresponding magnetic field distributions). (B) The conductivity images from multiple environments are used to train a deep neural network using the error backpropagation algorithm. The trained network accurately predicts the corresponding magnetic solution when environment conductivity images are shown.

matches the slope of the training curve, which means that the model is able to generalize well on the testing dataset.

We use a relative error metric $err_s(x, y)$ at the pixel level to assess the accuracy of the model for each sample input s:

$$err_s(x, y) = \frac{|\phi_s(x, y) - \hat{\phi}_s(x, y)|}{\phi_s(x, y)} \tag{3.2}$$

where $\hat{\phi}_s$ is the predicted magnetic field for the input sample s and ϕ_s is the ground truth generated with a commercial simulator. The average relative error per-pixel for a batch of n samples is defined as the mean value of the relative per-pixel error. In logarithmic scale (decibels):

$$err_{avg}(x, y) = \left. \frac{\sum_s err_s(x, y)}{n} \right|_{dB} \tag{3.3}$$

Fig. 3.2B shows the relative error of MaxConvNet for the testing dataset: less than -25dB for almost every pixel. This shows that the model is able to learn the relationship between magnetic field and conductivity, and is able to generalize to unseen samples. An attentive reader might notice that the average relative error around the center of the map is lower (i.e., below -35dB) than the error at its boundaries. This can be explained by the structure of our dataset, which places most of the obstacles far from the map boundaries, meaning that MaxConvNet is especially good at determining the magnetic fields around obstacles.

Fig. 3.2C shows the performance of MaxConvNet for three sample scenarios with increasing complexity. Fig. 3.2C.i is a simple case where the environment is free of obstacles. The relative error map (bottom) shows that MaxConvNet is able to reproduce the ground truth (top).

Fig. 3.2C.ii and Fig. 3.2C.iii present more complex cases where the environment is cluttered with obstacles. The error maps show an error rate less than -20dB for almost all pixels in the scenes. Overall, the model is able to learn the obstacle positions (i.e. with an error rate less that -80dB) and predict the correct magnetic field distribution.

3.2.1 Path Planning

To compute a feasible path from the predicted distribution, we have implemented Adam's algorithm [99]. Table 3.2 summarizes Adam's parameters. Assuming a robot acting in discrete time steps, at each time step the algorithm takes the gradient of the predicted field as well as the current position of the robot and returns a path to the goal (see Materials and Methods for details). The algorithm does not need any information about the goal since it is already encoded in the gradient information given as input.

Fig. 3.3 shows the computed path for some scenarios. The length of the path depends on the learning rate parameter that controls how much to change the position of the robot in response to the estimated gradient each time the path is updated towards the goal. Fig. 3.8 shows that the higher the learning rate the longer the path. An excessively large learning rate results in a coarse exploration of the gradient information and consequently generate a sub-optimal set of positions towards the goal. A too small learning rate could, in turn, lead to a long optimization process that might get stuck. To avoid either problems, we fixed the learning rate to 0.01.

Fig. 3.9 compares the paths computed on 4000 machine-predicted distributions and their ground truth in terms of path length. The ground truth path is produced from a strictly monotonic gradient [100], implying it is the optimal path [101]. The average ratio between optimal and the MaxConvNet-computed path length is $0.993 \pm$ 2.97e-05 which confirms the high accuracy of MaxConvNet in predicting the correct magnetic distributions. The main advantage of our approach is that it produces complete paths without local maxima, a problem present in other potential fields methods [18] that can cause a robot to get stuck. Fig. 3.3 shows some classic examples where traditional planners can get stuck, but where MaxConvNet can find a path to the goal location.

3.2.2 Physics-Based Simulations and Real-Word Experiments

We used MaxConvNet for the autonomous navigation of flying and wheeled robots. We used realistic models of the Parrot AR.Drone 2.0 quadcopter and of a sport utility vehicle (SUV) in a high-fidelity visual and physical simulator, AirSim [102]. Fig. 3.10 depicts the simulation setup. Fig. 3.4A and Video S1-2 shows two simulated scenarios where a quadcopter and an SUV, controlled by MaxConvNet, are making their way to a goal position while avoiding obstacles. Fig. 3.4A presents the reconstructed maps of the virtual environments.

Moreover, we have implemented a game-based scenario where an AI tries to catch a human-controlled player using MaxConvNet in a highly dynamic environment. The position of the player is encoded as the goal and used along with the obstacle information by the AI to predict a feasible path to track the player (see Video S3 for an example).

Furthermore, we used MaxConvNet running on a NVIDIA Jetson Nano computing board to control a Traxxas Stampede rover equipped with Ydlidar X4 lidar and a PixRacer autopilot (see Fig. 3.5). The autopilot uses GPS information along with its inertial measurement unit to estimate its current position. We run a set of experiments outdoors as shown in Fig. 3.4B and Video S4.

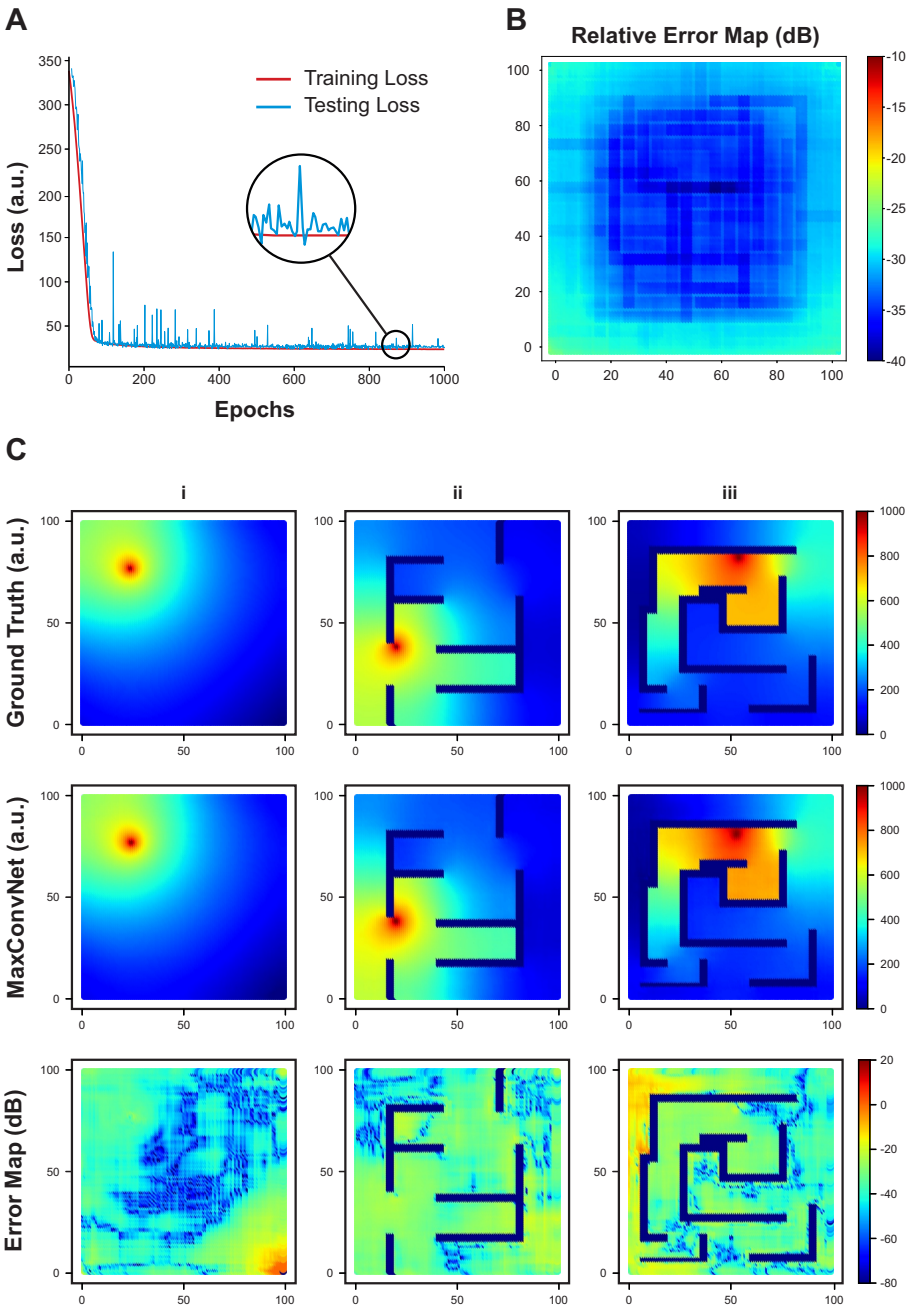


Figure 3.2 Performance of MaxConvNet: (A) We use the mean square error to train MaxConvNet. The testing loss decreases conjointly with the training loss. The network is able to generalize to unseen environments. (B) We calculate the average relative error per pixel for the testing set (4000 samples). The average relative error is under -25dB per pixel. (C i to iii) depict three environment scenarios with increasing complexity. MaxConvNet (middle) is able to reproduce the ground truth (top). The error map (bottom) shows the relative error per pixel (less than -20dB for almost all pixels in the map). The x-y axes of each sub-figure refer to the abstract positions of pixels in the maps i.e., in [0, 100] each dimension.

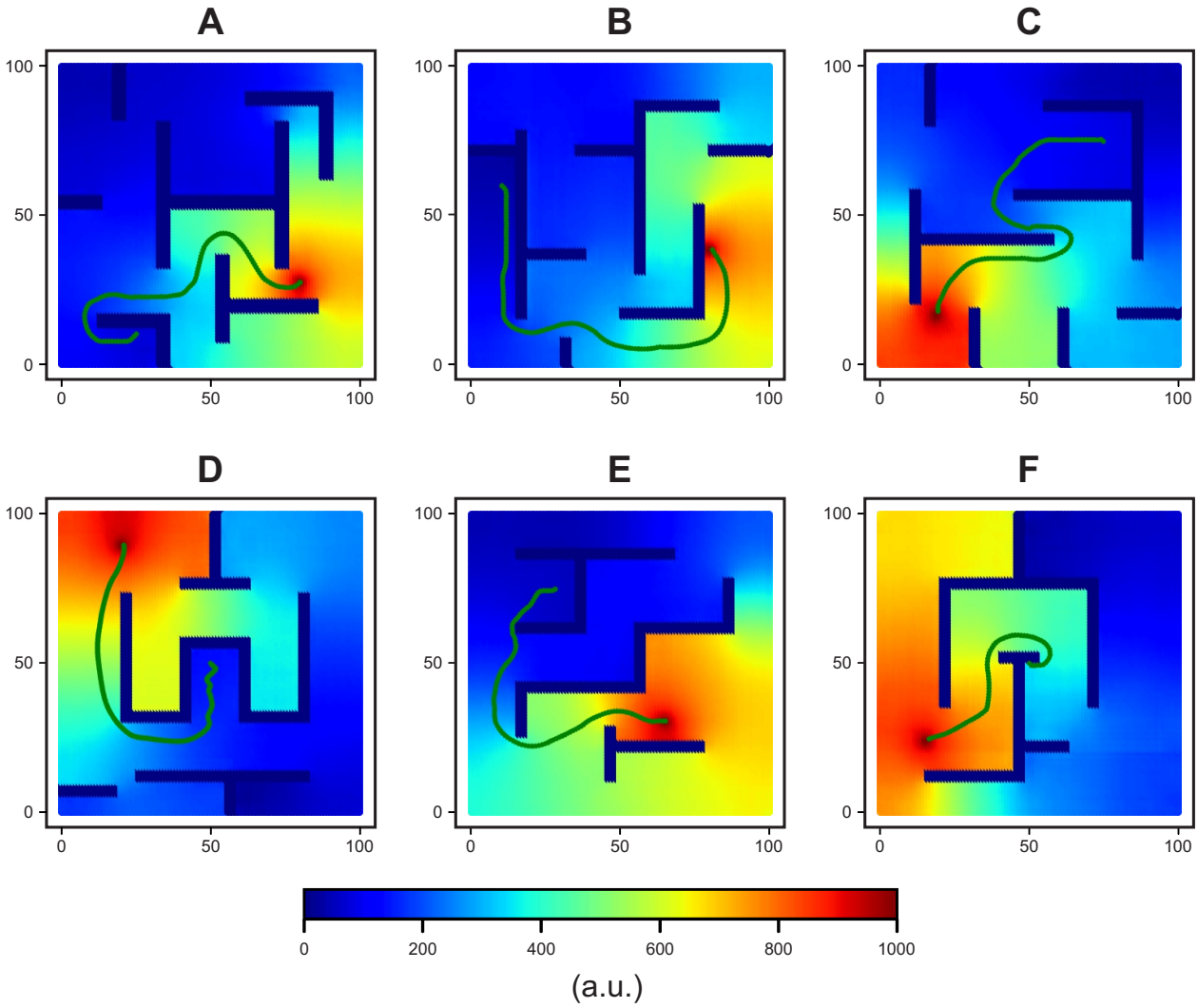


Figure 3.3 Gradient-based path: (A to F) The gradient information of machine-predicted magnetic fields can be used to compute a continuous path from a starting point to a goal position. The predicted magnetic distribution is agnostic to local maxima problems as it is based on Maxwell's equations for the magnetic field. The x-y axes of each sub-figure refer to the abstract positions of pixels in the maps i.e., in [0, 100] each dimension.

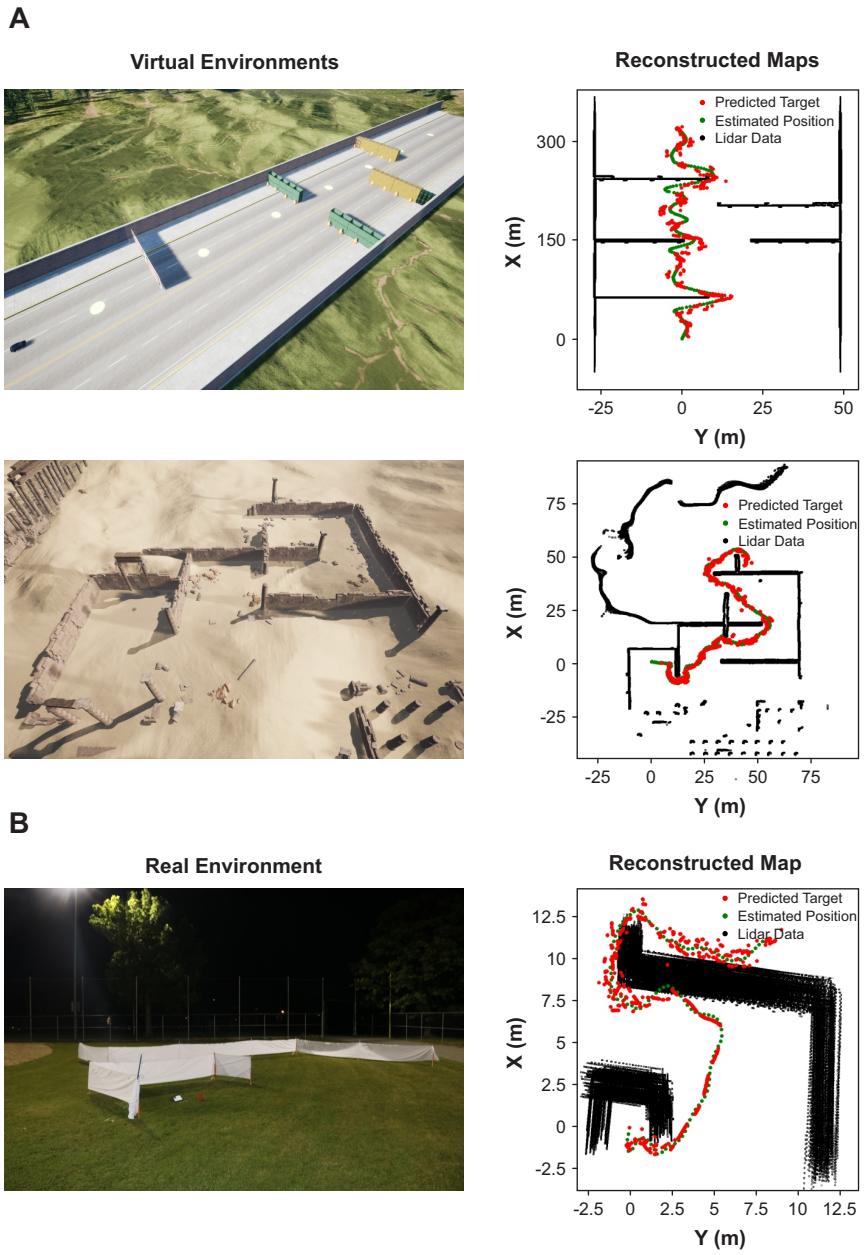


Figure 3.4 AirSim physics-based simulations and real-world experiments: (A) presents two virtual environments in AirSim: MaxConvNet predicts the next path to follow at 100 fps. The lidar data captured by the drone and the ground vehicle as well as the predicted path and the estimated positions has been superimposed to reconstruct the environments. (B) presents a real-world environment. MaxConvNet predicts the next path to follow at 7 fps (limited by the lidar performance). Obstacle data is collected using Ydlidar X4 lidar. Lidar data from different frames are superimposed along with predicted future positions and estimated positions to reconstruct the map.

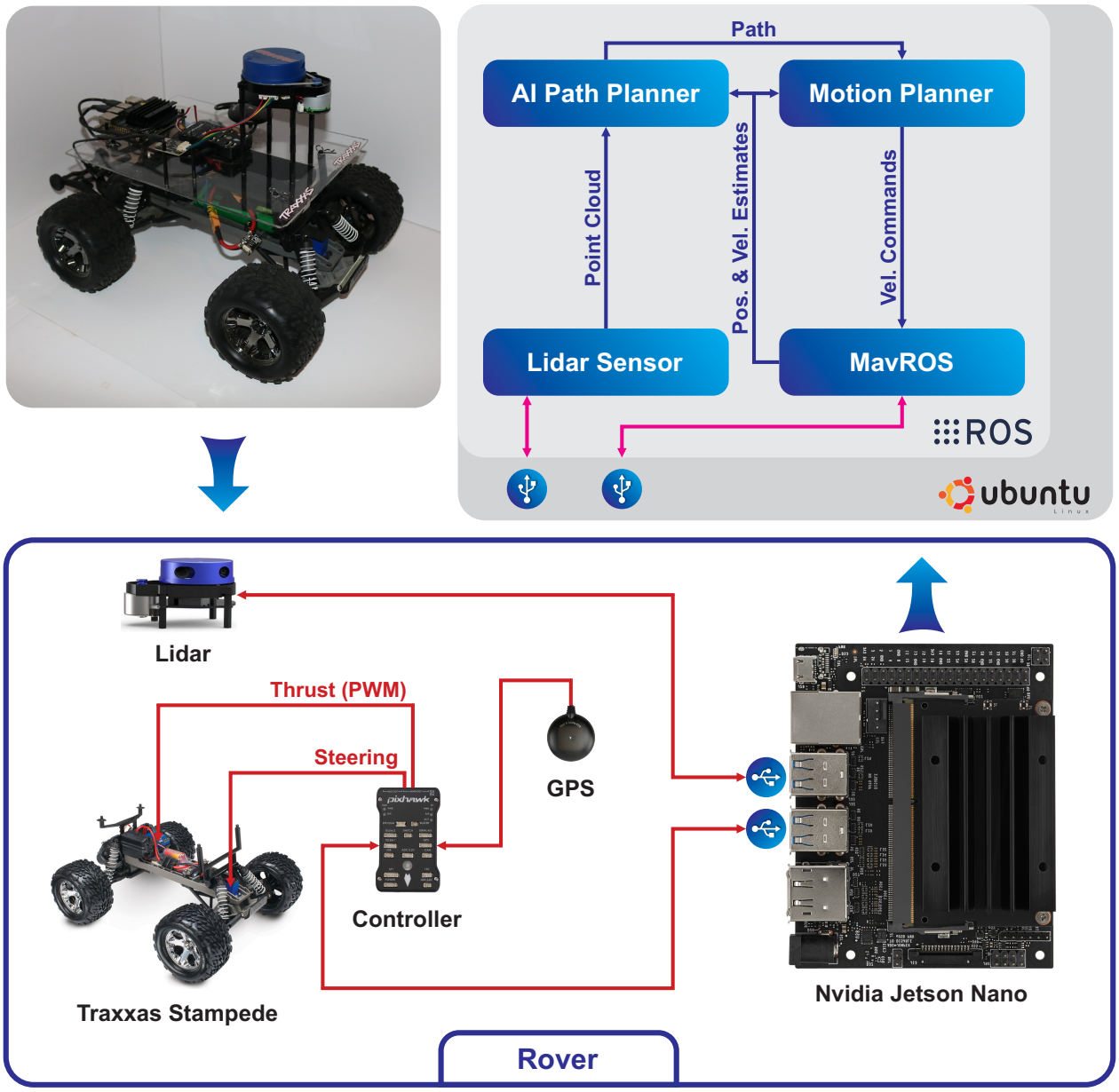


Figure 3.5 Experiments setup: We set up a wheeled platform using Traxxas stampede frame with a servo motor in the front of the vehicle for steering and a throttle motor at the back connected to the ESC (electronic speed control VXL). We used YdLidar X4 to collect 2D environment data and run our path planner on an NVIDIA Jetson Nano.

3.2.3 How Does MaxConvNet Compare to State-of-the-Art Path Planning?

We compare MaxConvNet and RRT* [23] in terms of path cost and runtime. Qualitatively, Figs. 3.6A and 3.6B show two scenarios where the path computed by MaxConvNet is shorter and smoother that the one produced by RRT*. The quality of the path generated by RRT-based approaches depends on the number of nodes (a parameter). The denser the tree, the better the path. However, this significantly slows down the algorithm. In Fig. 3.6A.ii, RRT* takes 30 seconds to compute a 125-meter path using 1000 nodes, while using 3000 nodes in Fig. 3.6A.iii leads to a path length of 103m, but at the expense of 217 seconds of computation time. For the same scenario, MaxConvNet was able to compute a shorter path in only 0.07 seconds, and similarly for Fig. 3.6B. Overall, MaxConvNet can compute a shorter path 400 times faster than RRT* with 1000 nodes.

Quantitatively, we compare RRT* (with 300 and 1000 nodes) and MaxConvNet over 10 different environments. Fig. 3.6C shows that the path length for MaxConvNet is *always* shorter than RRT* and Fig. 3.6D shows that the model has a constant, very high speed, 400 times faster than RRT* with 1000 nodes.

We can conclude that MaxConvNet is more efficient than state-of-the-art sampling-based approaches: it is much faster, finds near-optimal paths, has constant runtime, and its results do not depend on scene complexity or path length.

3.2.4 MaxConvNet Performance on Different Computers

We have conducted a statistical study to investigate the performance capabilities of MaxConvNet on different computers. We used a set of 1000 input environments on the most common Nvidia Graphical Processing Units GPUs–namely Jetson NANO, Jetson TX2, GTX 1050, GTX 1060, GTX 1080 Ti–, as well as on a Raspberry Pi 3 and a standard Intel i7-8750H. Table 3.1 depicts the minimum, maximum and mean values of the execution time of our model on the aforementioned platforms. Results show that MaxConvNet can run at up to 15 Hz on an embedded computer (the Jetson TX2) and up to 200 Hz on an advance GPU (the GTX 1080 Ti). It is worth noting that our model prototype is implemented in Python, and we expect better performance with a more optimized implementation with TensorFlow Lite [103].

3.3 Discussion

We propose an AI based approximator of Maxwell's equations that enables their use in applications with tight resource constraints. We also show how this new strategy can be applied to robots to perform navigation tasks. This application is of wide interest as it allows a low-cost navigation

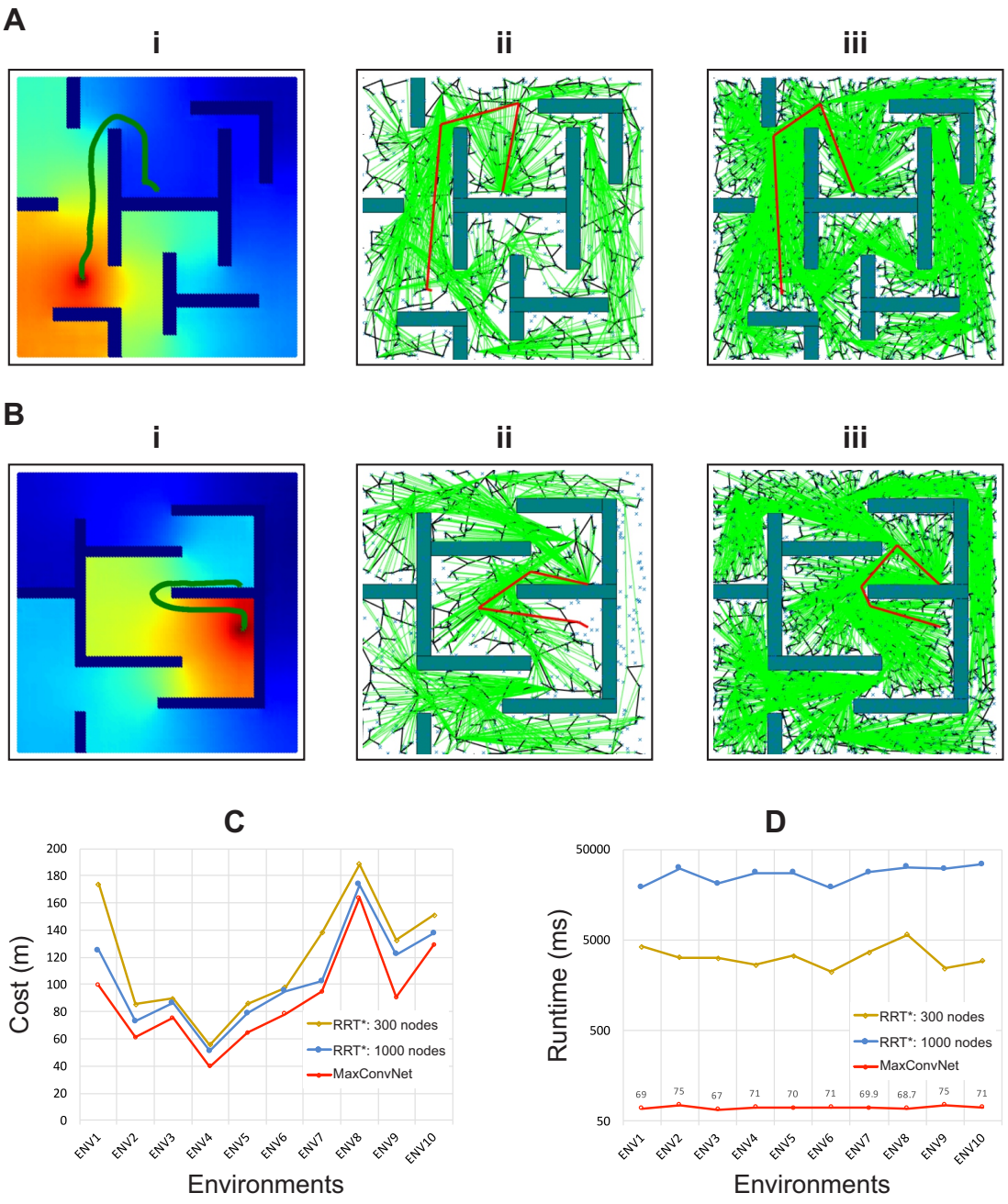


Figure 3.6 MaxConvNet vs RRT*: (A i) MaxConvNet runtime = 0.07s, cost = 99.27m, (A ii) RRT* number of nodes 1000, runtime = 30.12s, cost 122.01m, (A iii) RRT* number of nodes 3000, runtime = 217s, cost 111.33m, (B i) MaxConvNet runtime = 0.07s cost 75.17m, (B ii) RRT* number of nodes 1000, runtime = 27.45s, cost 104.24m, (B iii) RRT* number of nodes 3000, runtime = 329s, cost 84.64m. We compared MaxConvNet and RRT* in 10 different environment (C) Cost comparison, (D) Runtime comparison.

Table 3.1 MaxConvNet execution time on different Platforms.

		Nvidia GPUs					**CPUs**	
		Jetson Nano	Jetson TX2	GeForce GTX1050	GeForce GTX 1060	GeForce GTX 1080 Ti	RPI 3	Intel i7
MaxConvNet Execution Time (s)	**Min**	0.0920	0.0455	0.0132	0.0086	0.0045	1.901	0.039
	Max	0.1306	0.298	0.0199	0.019	0.006	2.885	0.081
	Mean	0.1065	0.074	0.0164	0.0093	0.005	1.982	0.033

in a highly dynamic and unknown environment, and ensures optimal paths in complex real world scenarios with static and dynamic obstacles.

MaxConvNet works by automatically recognizing the distribution of a virtual magnetic field in a 2D environment defined as a conductivity map. On top of MaxConvNet, we use an optimization algorithm to compute a feasible path given the gradient information of the magnetic distribution. The attentive reader might have realized that the path generated by MaxConvNet does not take into account the kinematic constraints of the robot, but these can be easily taken into account when executing the optimization algorithm for the generation of the path.

Results show that MaxConvNet makes correct predictions and avoids the problem of local maxima. In addition, MaxConvNet outperforms state-of-the-art navigation algorithms by ensuring completeness and providing an optimal path in real time. Additionally, the present method can be readily extended to other navigation problems, such as flocking (see Video S5) or pursuit, as we demonstrate with a gaming example (Video S4).

The neural network presented in this paper can be an inspiration to understand how magnetic navigation is performed by living creatures, or how the problem of visual path-planning in a dynamic, unknown environment can be solved efficiently by a neural architecture, given a simple model of the environment. Most importantly, the environment model and conductivity could be modified to account for the *dynamics* of the robot being considered to achieve specific trajectories such as those needed for drone or car racing. These implicit optimizations would come at *no performance cost*.

The low computational cost of MaxConvNet allows implementations on small, low-cost, single-board computers as the Jetson Nano or the Raspberry Pi, paving the way for pervasive robot appli-

cations. In addition, one can train the system using *exclusively* simulation data from well known physical phenomena leading to very straightforward implementation on physical robots.

We believe the next steps beyond this proof-of-concept are clear: MaxConvNet can be extended to 3D to enable a more efficient navigation for flying and under-water robots. Fig. 3.11(a) shows the magnetic field distribution of a 3D environment computed using our physics-based model. Fig. 3.11(b) depicts the gradient information that can be used to compute a free path from any starting position to the goal. In addition, the network can be trained with an arbitrarily large number of environments data with different obstacle shapes. The performance can be further improved increasing the environment resolution (i.e. 512x512).

3.4 Materials and Methods

3.4.1 Physics-Based Model

Consider an arena in which the magnetic field is given by a vector field $\vec{B}$. At each point $p(x, y)$ we have $\vec{B}(x, y)$ and its gradient $\vec{G}(x, y)$. $\vec{G}$ is a vector that points in the direction in which $\vec{B}$ rises most quickly and its magnitude determines how fast the field rises in that direction. The gradient information can be used to optimally connect any point in the environment to a goal.

In a cluttered environment with obstacles, the goal is assigned to the highest conductivity whereas the obstacles are assigned to zero conductivity. The environment is assigned to a degraded conductivity which fades away when moving away from the goal (see equation 1). Electrical currents are assumed to be floating in the environment, and the magnetic field induced by these currents is used to find a free and optimal path to the goal.

The magnetic field can be modeled with Maxwell's equations:

$$\frac{\partial \vec{B}}{\partial t} = \nabla \times \vec{E} \tag{3.4}$$

$$\nabla \times \vec{B} = \mu_0 \vec{J} + \mu_0 \frac{\partial \vec{D}}{\partial t} \tag{3.5}$$

$$\nabla \cdot \vec{B} = 0 \tag{3.6}$$

where $\vec{B}$ is the magnetic field, $\vec{E}$ is the electrical field, $\vec{J}$ is the density of electrical currents and μ_0 is the free space permeability. Equation 3.4 (Maxwell-Faraday) draws the relationship between the magnetic field $\vec{B}$ and the electrical field $\vec{E}$. This relationship states that a time-varying magnetic field will always coexist with a spatially varying, non-conservative electric field, and vice-versa. Maxwell-Ampere's equation (Equation 3.5) states that a magnetic field $\vec{B}$ can be generated in two

ways: i) by electrical currents as expressed by Ampere's law (see Equation 3.7) and ii) by changing electric fields (a.k.a., displacement currents $\vec{D}$) generated by the magnetic field induced by the floating currents $\vec{J}$. Under a low frequency ω = 0.05, it is safe to neglect displacement currents. Hence, Equation 3.5 can be then reduced to Equation 3.7:

$$\nabla \times \vec{B} = \mu_0 \vec{J} \tag{3.7}$$

According to Ohm's law, the current density $\vec{J}$ is:

$$\vec{J} = \sigma \vec{E} \tag{3.8}$$

where σ is the electrical conductivity. The magnetic field $\vec{B}$ is usually an alternating field and can be expressed in a time-dependent form as:

$$\vec{B} = \vec{B_0} \cdot e^{i\omega t} \tag{3.9}$$

By applying Ohm's Law (Equation 3.8) and Faraday-Maxwell (Equation 3.4), Equation 3.7 can be written as:

$$\nabla \times \nabla \times \vec{B} = -k^2 \vec{B}. \tag{3.10}$$

where $k^2 = i\omega\mu_0\sigma$, defined for the goal, obstacles and the free environment as in Table 3.3. Gauss' law for magnetism (Equation 3.6) asserts that the net outflow of the magnetic field through any closed surface is zero. Using Gauss' law along with the vector calculus identity relationship $\nabla \times (\nabla \times \vec{B}) = \nabla(\nabla \cdot \vec{B}) - \nabla^2 \vec{B}$, Equation 3.10 reduces to:

$$\nabla^2 \vec{B} = -k^2 \vec{B} \tag{3.11}$$

3.4.2 Dataset

We implemented Equation 3.11 in COMSOL[1] and solved using its finite element solver. We used COMSOL LiveLink for MATLAB[2] to automate the dataset collection. Using MATLAB, we have randomly generated obstacles in a square arena (10x10m). The geometrical properties of the arena, obstacles and goal are depicted in Table 3.3. The generated scene is sent to COMSOL to compute the distribution of the magnetic field, which is then saved to disk. Fig. 3.1A shows an example of a magnetic field computed by COMSOL given a conductivity distribution. A dedicated script was implemented to process COMSOL's output and store the scene as an image-like array where each

[1] https://www.comsol.com/

[2] https://www.comsol.com/release/5.4/livelink-matlab

pixel corresponds to a magnetic field value.

We build a dataset with 20,000 samples. Each sample is an image-like array with two channels: i) the scene conductivity distribution and ii) the corresponding magnetic field distribution. The size of each sample is 101x101. This choice allows a fast training and is suitable for both indoor and outdoor use. For an indoor application, we can cover an area of 10x10m with a step resolution of 0.1m. For an outdoor application, we can use the same model to process 100x100m with a resolution of 1m. Furthermore, this choice makes us able to show the applicability of our approach with limited GPU resources. However, for industrial use, more advanced GPUs can be leveraged to train the model with finer resolution.

3.4.3 Optimization and Training

The internal structure of MaxConvNet is shown in the Fig. 3.7 and Table 3.4. MaxConvNet is implemented as a convolutional auto-encoder trained in a supervised manner. In the encoder stage, the conductivity map of a single environment is processed by multiple layers of convolution, max pooling, non linearity and batch normalization. In particular, the encoder stage consists in five blocks of layers whereby each convolutional layer is followed by a pooling layer, a rectified linear unit (ReLU) [104] and batch normalization [105].

In the decoder stage, the received data is iteratively up-sampled using nearest neighbors [106] and processed using convolutions followed by non-linearity and batch normalization layers. To avoid overfitting, we use dropout [107] and $\mathcal{L}2$ regularization [108]. Dropout randomly selects some nodes and removes them along with all of their incoming and outgoing connections according to an hyperprameter which defines the rate of dropping. L2 regularization updates the general loss function by adding a regularization term. The cost function with the regularization term is:

$$Loss_{MSE} = \frac{(\phi - \hat{\phi})^2}{n} + \frac{\lambda}{2n} \sum_{n} w \tag{3.12}$$

where ϕ is the predicted potential, $\hat{\phi}$ is the potential computed by COMSOL, λ is a hyperparameter, n is the size of the training batch and w is weights of the network.

The learnable parameters of MaxConvNet are iteratively adjusted with the error backpropagation algorithm [109]. The neural network is trained by minimizing the sum of squared errors loss between COMSOL-computed and predicted magnetic field distributions. The use of a CNN allows us to considerably reduce the number of learnable parameters using localized and shared receptive field structures. This allows faster training compared to a conventional deep neural network. We use the Adam optimizer [99] to adjust the network parameters.

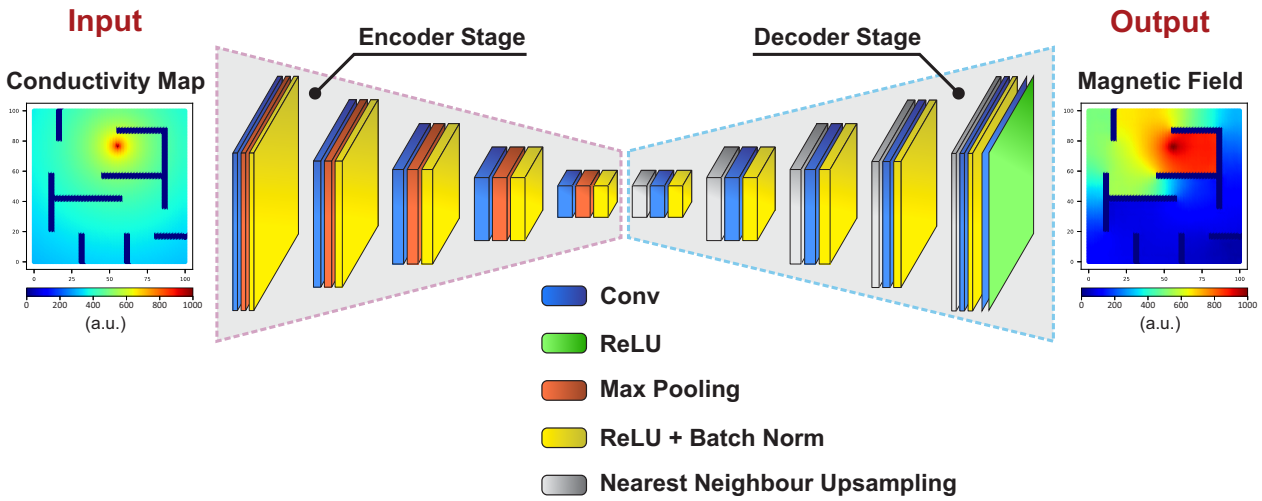


Figure 3.7 Architecture of MaxConvNet: When a conductivity image of an environment is taken as input, the network first processes the image through an encoder stage (five rounds of convolution, ReLU nonlinearity, and pooling layers). Then, a decoder stage follows with 5 rounds of up sampling, convolution, and ReLU nonlinearity. Dropout and L2 regularization are used to avoid overfitting. We compute ground truth magnetic field distributions using a commercial simulator (COMSOL) to obtain the loss function. See Table 3.4 for details on the architecture.

We initialize the weights using the method described in [110]. We implement the network in Tensorflow [111] and we train on an NVIDIA GTX Gforce 1080 TI GPU (11GB) with a batch size of 50 samples.

3.4.4 Path Following

The proposed planner generates a path as a set of points connected by straight lines. To navigate the path, a robot shall move from point to point until it reaches the end of the goal. We implement a modified version of Craig Reynold's path-following algorithm [112] to ensure smooth navigation. Algorithm 1 outlines the different steps of the steering behavior: given its current position and velocity, a robot estimates its future location and projects it on all the path's segments. The robot sets its next target to the closest projected point, also called the normal point. If all the normal points fall outside the path segments then the robot sets its next target to the l^{th} closest normal point where l is a fixed offset. Given the next target, the robot computes the velocity command if the distance between the next target and the predicted future location is greater than the path radius. With a smaller radius r, the robot has to follow the path more closely; a wider radius allows it to stray more. As it moves, the robot strips the traveled distance from the path to avoid oscillatory behavior.

Algorithm 1: Modified Version of Craig Reynold's path-following algorithm.

Input: A path $\mathcal{P} = \{\psi_1, \psi_2, ..., \psi_n\}$, where ψ_i is a line segment *s.t.* $\psi_i = [\alpha_i \beta_i]$, α_i is the segment's start point and β_i is the segment's end point, n is the number of segments in the path and r is the path radius.

Output: Velocity command $\vec{V}^{(t+1)}$

Get current position $p^{(t)}$;
Get current velocity $\vec{V}^{(t)}$;
Predict vehicule's future location $\hat{p}^{(t+1)} \leftarrow f(p^{(t)}, \vec{V}^{(t)})$;
for ψ_i *in* $\mathcal{P}$ **do**
    Get projection $p_i^{\perp}$ of $\hat{p}^{(t+1)}$ on ψ_i ;
    if $p_i^{\perp} \notin \psi_i$ **then**
        $p_i^{\perp} \leftarrow \beta_i$;
    end
    Calculate $d_i = \| \overrightarrow{\hat{p}^{(t+1)} p_i^{\perp}} \|$;
end
if $p_i^{\perp} \notin \psi_i \forall i \in [0..n]$ **then**
    Set target position $p^{(t+1)} = p_{k+l}^{\perp}$ *s.t.* $k = \text{argmin}_{i \in [1..n]}\, d_i$ where l is a fixed offset;
else
    Set target position $p^{(t+1)} = p_k^{\perp}$ *s.t.* $k = \text{argmin}_{i \in [1..n]}\, d_i$;
end
if $\| \overrightarrow{\hat{p}^{(t+1)} p_i^{\perp}} \| > r$ **then**
    calculate velocity command $\vec{V}^{(t+1)} = \overrightarrow{p^{(t)} p^{t+1}}$;
end
Strip the travelled distance from the current path $\mathcal{P}$

3.5 Supplementary Materials

This section includes supplementary figures, tables and videos'captions used to support the findings of this chapter.

3.5.1 Supplementary Figures

Fig. 3.8 shows the impact of Adam's learning rate on the path length. The higher the learning rate the longer the path.

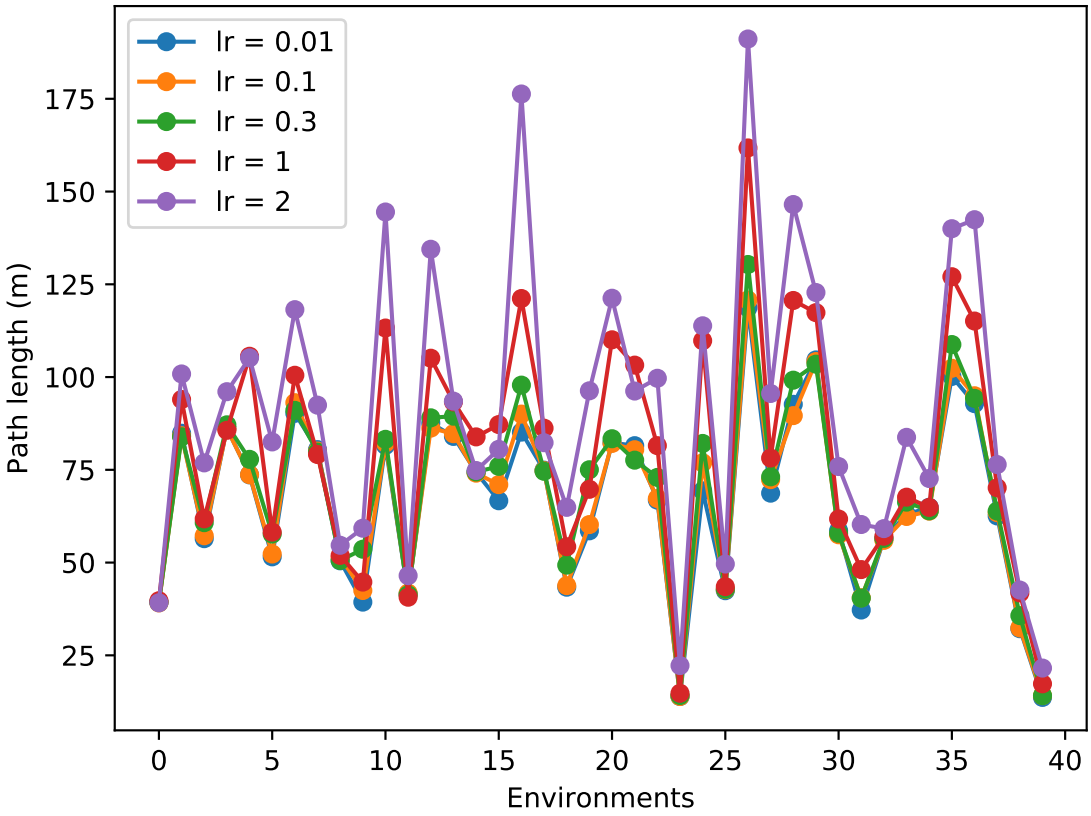
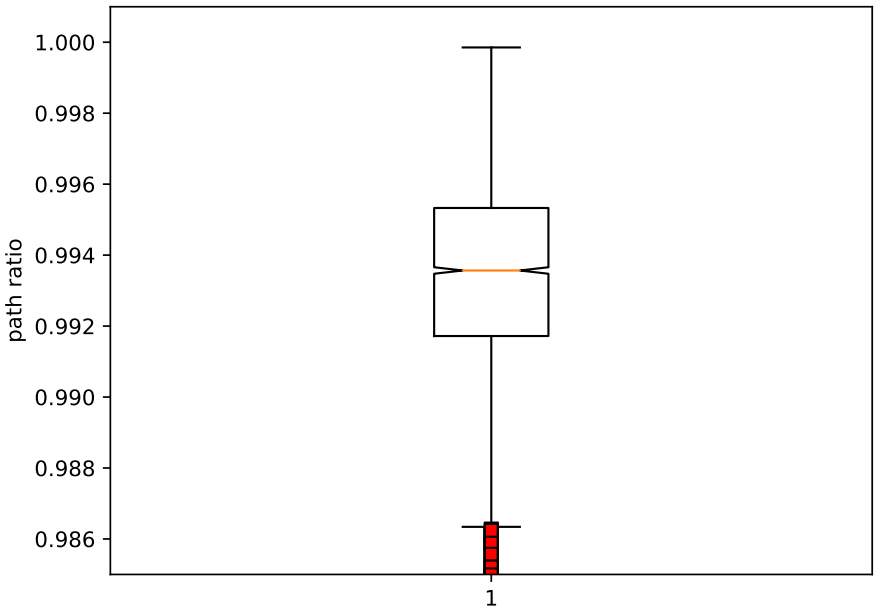
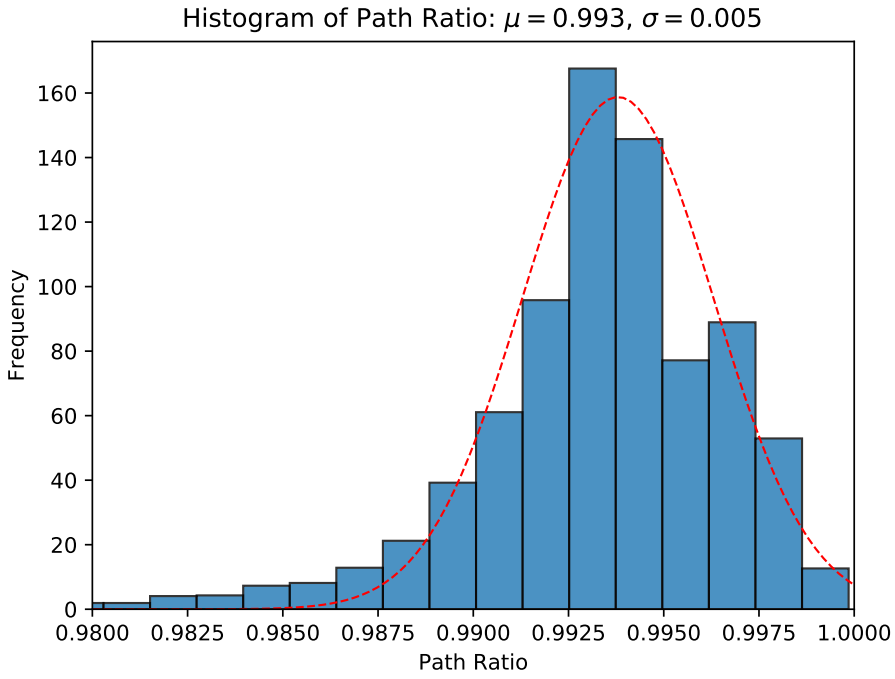


Figure 3.8 Impact of Adam's learning rate on the path length.

Fig. 3.9 depicts the ratio between the path computed using the real magnetic distribution and the one computed using the machine predicted distribution: Fig. 3.9(a) and Fig. 3.9(b) are two different visualizations: The length ratio variable follows a Normal Distribution $N(\mu = 0.993, \sigma = 0.005)$, The median value is 0.993.



(a)



(b)

Figure 3.9 Path ratio

Fig. 3.10 depicts AirSim physics-based simulation: AirSim server computes the physics of the vehicle model and simulate the sensors and the environment to mimic real-world behaviors (such as sensor drift, position errors). AirSim is plugin for Unreal Engine 4 that enables photo-realistic rendering and perception data collection.On the client side, MaxConvNet runs inside the AI path planner module at 100 fps. The AI path planner requests lidar data and estimated kinematics from the server and produces a feasible path. The motion planner converts the computed path into velocity commands using a modified version of Reynold's algorithms.

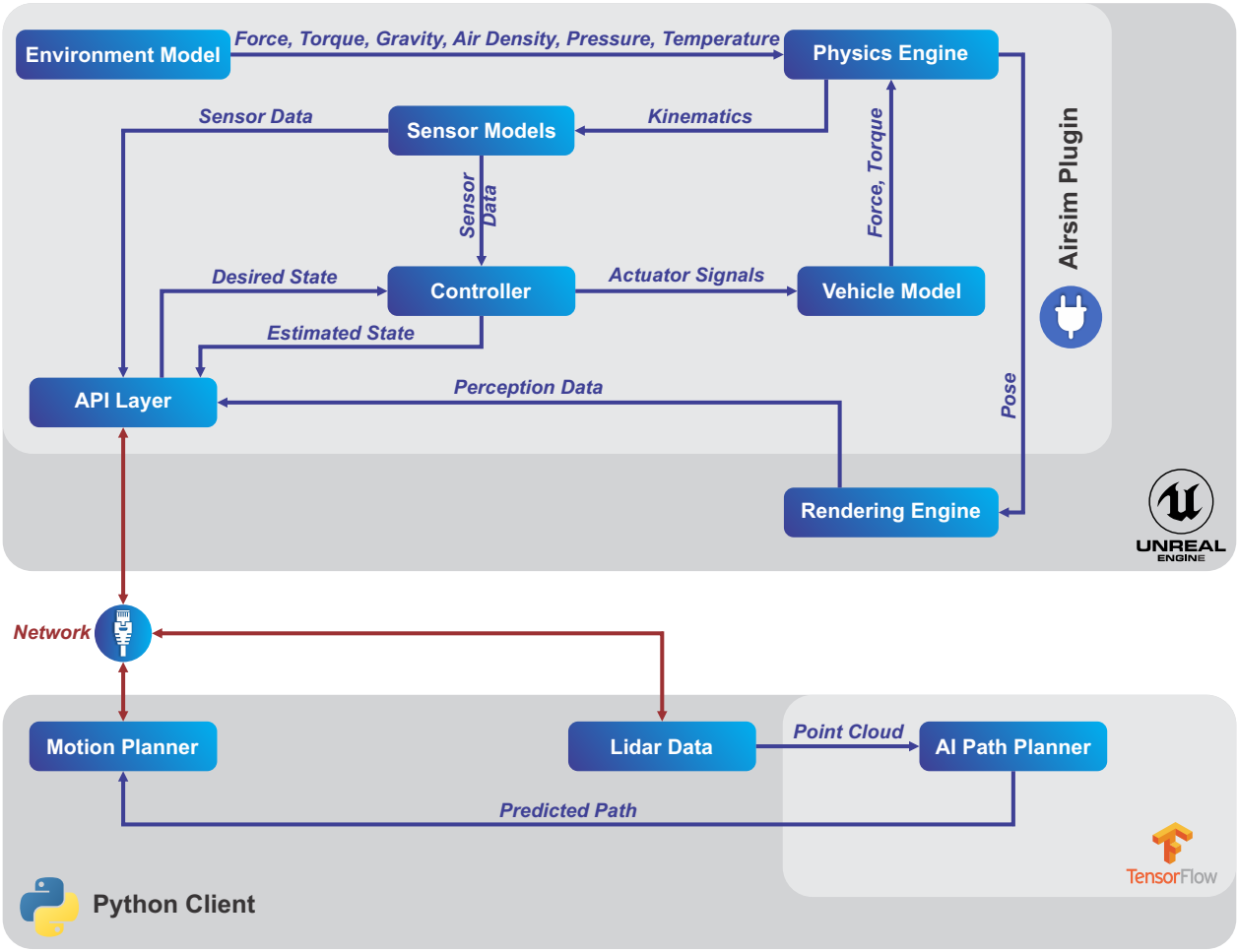
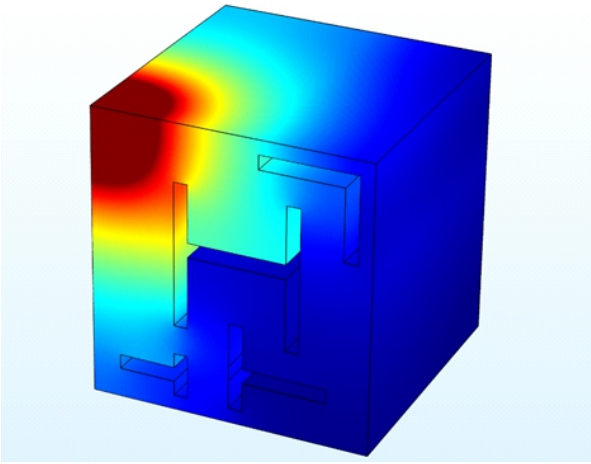
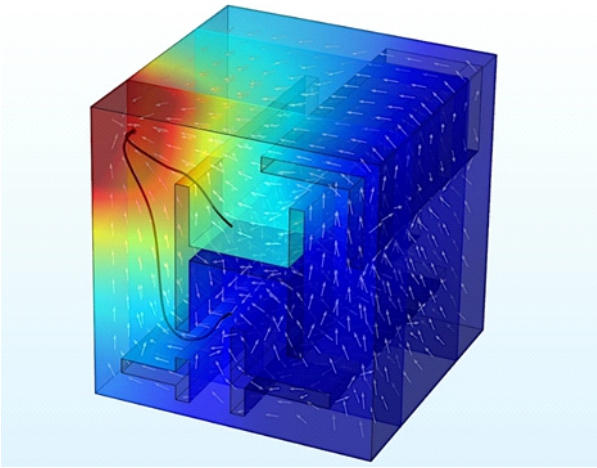


Figure 3.10 AirSim physics-based simulation.

Fig. 3.11 promotes to the eventual extension of our model to handle 3D environments: Fig. 3.11(a) shows the magnetic field distribution computed in a 3D environment using our physics model. Fig. 3.11(b) depicts the gradient information computed based on the 3D distribution shown in Fig. 3.11(a). A path is computed for two different locations to the goal (in red). This shows our approach can be easily extended to handle 3D environments.



(a) The gradient of the magnetic distribution.



(b) Magnetic field distribution computed using the physics model.

Figure 3.11 Magnetic field distribution computed in a 3D environment.

3.5.2 Supplementary Tables

Table 3.2 reports the parameters of Adam's algorithm used to compute an optimized path to the goal.

Table 3.2 Adam's parameters.

Adam's Parameters	Definitions	Values
lr	Learning rate	[0.01,1]
β_1	Exponential decay rate for the first moment estimates	0.9
β_2	Exponential decay rate for the second moment estimates	0.999
ϵ	Used to prevent division by zero	10e-8

Table 3.3 gives a detailed description to physics model: the physical properties of the different mediums in the model (goal, obstacles and free environment) and their geometrical properties including their shapes, sizes and the meshing.

Table 3.3 Physics model.

	Parameters		Definitions	Values
Physical Properties	σ_g		Goal conductivity constant	1e6
	σ_o		Obstacles conductivity constant	0
	σ_e		Environment conductivity constant	100
	μ_0		Free space permeability	12.56e-7
	ω		Frequency	0.05
	K^2	K_g^2	Goal wave number	$\mu_0 * \sigma_g * \omega * i$
		K_o^2	Obstacle wave number	$\mu_0 * \sigma_o * \omega * i$
		K_e^2	Environment wave number	$\mu_0 * \sigma_e * \omega * i$
Geometrical Properties	Shape_g/size_g		Goal shape/size	Circle/radius=0.09m
	Shape_o/size_o		Obstacles shape/size	Random/Random
	Shape_e/size_e		Environment shape/size	Square/side=10m
	(x_g,y_g)		Goal local coordinates	([0,10], [0,10])
	Meshing	Type	Type of the mesh element	Triangular
		Hmin/Hmax	Approximate size of the mesh element	0.3/0.5m

Table 3.4 is a detailed display of MaxConvNet architecture: Conv is a convolutional layer with a specified filter size, stride and number of filters. Conv1-5 are in the encoder stage, whereas Conv6-11 are in decoder stage. The Batch Norm column indicates whether Conv is followed by a Batch Normalization layer. The Nonlinearity column shows whether and what nonlinearity layer is used (preceding the Batch Norm if Batch Norm is used). NearestUpSample module allows to resize the input according to the Up_Factor parameter. Pooling is used for subsampling and nearest neighbor filtering for up sampling.

Table 3.4 MaxConvNet architecture.

Module Name	Filter size	#Filters/Channels	Stride/Up_Factor	Dropout	Pooling Size/stride	Batch Norm	Non-Linearity
Conv1	11x11	32	1x1/-	0.1	1x1/2x2	Y	ReLU
Conv2	7x7	64	1x1/-	0.1	1x1/2x2	Y	ReLU
Conv3	5x5	128	1x1/-	0.1	1x1/2x2	Y	ReLU
Conv4	3x3	256	1x1/-	0.1	1x1/2x2	Y	ReLU
Conv5	3x3	512	1x1/-	0.1	1x1/2x2	Y	ReLU
NearestUpSample	-		-/7		-		-
Conv6	3x3	512	1x1/-	0.1	-	Y	ReLU
NearestUpSample	-		-/13		-		-
Conv7	3x3	256	1x1/-	0.1	-	Y	ReLU
NearestUpSample	-		-/26		-		-
Conv8	3x3	128	1x1/-	0.1	-	Y	ReLU
NearestUpSample	-		-/51		-		-
Conv9	3x3	64	1x1/-	0.1	-	Y	ReLU
NearestUpSample	-		-/101		-		-
Conv10	3x3	32	1x1/-	0.1	-	Y	ReLU
Conv11	3x3	1	1x1/-	-	-	-	ReLU

3.5.3 Videos Captions

Movie S1. MaxConvNet controls a Parrot AR.Drone 2.0 quadcopter to fly to a goal position (green tag) at 5 m/s. We used AirSim to simulate the physics of the drone and UE4 to render the drone behavior in a realistic complex environment. MaxConvNet magnetic distribution as well as the generated path are rendered on the video as a reference.

Movie S2. MaxConvNet controls a Sport Utility Vehicle (SUV) across multiple goal positions (green tags) at 11 m/s. We used AirSim to simulate the physics of the car and UE4 to render the car behavior and the navigation environment. It is worth noting that motion control is out of the paper scope even though it can be easily integrated taking into consideration the kinematic constraints of the car.

Movie S3. We present a game scenario where an AI player tries to catch a human-controlled player using MaxConvNet in a highly dynamic environment. We developed the game using a client/server architecture. The communication is handled via UDP. The server side is implemented using UE4. We implemented a lidar-like sensor (using ray tracing) to collect perception data. The client, running MaxConvNet, connects to the server, requests for the AI player position, the goal position (Human-controlled player position) and lidar data. It makes a path prediction and sends it back to the server. The AI player running on the server side just follows the path.

Movie S4. We use MaxConvNet running on a NVIDIA Jetson Nano computing board to control a Traxxas Stampede rover equipped with a Ydlidar X4 lidar and a PixRacer autopilot. The autopilot uses GPS information along with its inertial measurement unit to estimate its current position.

Movie S5. We use AirSim to control the motion of 5 drones simultaneously using MaxConvNet. This video shows the applicability of our approach to multi-agent systems.

3.6 Acknowledgments

We would like to thank Aymen Soussia for the technical support. Funding: This work was supported by funding from the Natural Sciences and Engineering Research Council of Canada (grants STPGP 479149 and RGPIN 05165). Data and materials availability: All data needed to evaluate the conclusions in the paper are present in the paper and/or the Supplementary Materials. Additional data related to this paper may be requested from the authors.

CHAPTER 4 ARTICLE 2 - ON THE ROBUSTNESS OF CONSENSUS-BASED BEHAVIORS FOR ROBOT SWARMS

Preface

This paper gives insights as to how formally evaluate and verify the robustness and the resilience of consensus-based swarm behaviors. Our proposed model is based on formal methods which are known by their effectiveness and accuracy in checking real-time systems. More precisely, we propose a stochastic model-checking approach to verify the consensus based strategies for a networked robotic swarm in terms of the communication quality and robot failure probability.
The model is tested and validated on two common scenarios (leader election and auction-based task allocation), with satisfactory results: The assessment of the communication quality impact was comparable to what was recorded using physics-based simulations and real-world experiments. We further show that the model performs well in terms of verification time and memory usage on a standard computer.

Authors Majda Moussa and Giovanni Beltrame
Submitted to: Swarm Intelligence

In swarm robotics, behaviors requiring consensus, meaning having the robots agree on a set of variables, have attracted great attention over the years. Determining the robustness and applicability of these behaviors in harsh communication environments is an open area of research. In this paper, we propose the use of a formal software engineering technique, Statistical Model Checking (SMC), to model and assess the robustness of consensus-based behaviors from a communication standpoint. We validate our approach on two common scenarios for a robot swarm: the election of a leader, and the allocation of a set of tasks. With the proposed model, we verify the functional correctness of these consensus-based algorithms, as well as assessing their robustness to communication loss and robot failures.

4.1 Introduction

During the last decade, we have witnessed a rapid growth in the deployment of autonomous and robotic systems. Their application is continuously expanding in various critical domains such as healthcare, logistics, surveillance, and disaster relief. In this context, verifying the robustness and

resilience of swarm systems is of paramount importance. Whilst it is common practice to implement, understand, and debug the behavior of an individual robot, it is considerably harder to predict and guarantee the collective behavior of a swarm, especially when deployed in a dynamic, potentially hostile and unknown environment.

The fundamental characteristics of a swarm robotic system [43] are i) autonomy, *i.e.* the agents are able to interact with their environment with a high degree of independence, ii) decentralization, *i.e.* robots make local decisions using independent sensing and communication capabilities, and iii) robustness, *i.e.* the swarm can still perform its tasks even in case of multiple robot failures.

The scientific community invested a huge effort to provide advanced behaviors for robotic swarms conforming to these characteristics [113, 114]. These algorithms have been usually tested and validated using physics-based simulations and robot experiments (an overview of robotics validation methods can be found in [43]). These two techniques are limited by the number of experiments and the statistical significance of the results. As an alternative, formal methods (such as those commonly used in software engineering, as described by [115]) provide a mathematical model which can verify the outcomes (including the case of failures) of the system within given confidence bounds. Unfortunately, few works (listed in Section 4.2) use this approach for the evaluation of the robustness of robotic swarms as critical systems. In this paper, we try to bridge this gap by providing a methodology for the formal modelling of consensus-based algorithms.

[66] presented a generic consensus system named Virtual Stigmergy (VS). The VS provides a way for a swarm of robots to agree on a set of key-value pairs, and can be used to coordinate the behavior of the swarm. [66] showed the applicability of the VS using simulations with thousands of robots and different packet drop rates. However, the work provided no formal model for the VS, and the relationship between communication reliability, convergence, and time to convergence could hardly be inferred from the experimental results.

In this paper, we build a new formal model of the VS to ensure its resilience and robustness to packet drop probability and robot failure probability. We then validate the model in the context of consensus-based behaviors. First, we analyze a common scenario where the robots must agree on the highest (or the lowest) value of a given parameter – *i.e.* electing one of the robots in the swarm as a leader –. Furthermore, we show the applicability of our model to a more complex scenario, the partitioning of a set of given tasks based on a bidding approach [116].

To the best of our knowledge, this is the first attempt to formally assess and quantify the correctness and robustness of consensus-based behaviors for robot swarms. While this problem is still relatively unexplored in literature, a notable exception is the work of [117] which investigates a protocol allowing the swarm to reach consensus over a certain opinion. However, they only qualitatively verify the correctness of the design without assessing the impact of the communication and

robot reliability on the studied protocol. A later study [75] proposes a framework to evaluate the faulty tolerance of Multi-agent Systems (MAS). Although the approach is promising, it does not give insights on the time to reach consensus. Furthermore, it only considers temporal epistemic properties which account only for certain knowledge acquired by agents during their interactions. No attempt was made to model and verify uncertainty, which is an important feature in a MAS. In this work, we addressed the qualitative and quantitative aspects of the system, while taking into consideration the impact of robots' failure using well-defined, measurable properties to assess system requirements. This enables an efficient simulation of some system dynamics assuming that robots might fail is equivalent to robots going out of communication range. Under this assumption, we were able to raise up the abstraction level of the model by ignoring the exact position of the robots which makes the model more tractable when the number of robots significantly grows.

This paper is structured as follows: further related work is discussed in Section 4.2; Section 4.3 presents an overview of the SMC technique and our motivation for its use; in Section 4.4, we describe our formal model and show how we use it to probabilistically verify the behavior of the swarm; Section 4.6 is dedicated to study the performance of the model against experimental and simulation approaches; finally, Section 4.7 draws some concluding remarks.

4.2 Related Work

Swarm robotic systems have been used in a wide range of applications with clear requirements for robustness, resilience and flexibility [61, 62]. The collective behavior of swarm systems might be subject to arbitrary faults such as Byzantine Generals problem [63] which may completely paralyze their coordination mechanism and prevent them making a collective agreement *i.e.* consensus.

Researchers from many institutions have proposed different approaches and methodologies implementing fault-tolerant behaviors that require a formal notion of consensus *e.g.,* task allocation [64], formation control [65], leader election [66], and determination of coordination-based variables, *e.g.,* rendezvous time [67].

Those consensus-based behaviors rely on a coordination system (*e.g.,* stigmergy [50, 60]) to share data in the swarm and ensure convergence. [66] have proposed a new use of the stigmergy concept under the name of virtual stigmergy, which allows to share a collection of (key, value) pairs. Whenever an agent reads or makes an update to a value assigned to a given key, it exchanges the data with its neighbors. Consensus is reached when every agent in the swarm shares the same information.

In general, authors show the applicability of their approaches using simulations, experiments, or models that do not consider the presence of faults. Overall, there are few studies that formally focus on the robustness of consensus.

The initial work of [118] proposed to study the eminent alternative to experiments and simulations, that is the formal verification, and particularly Model Checking (MC). This technique allows to construct a mathematical model of all the possible behaviors of the system. The model is then assessed against a logic formula representing a required functionality of the system. [70] suggested an automated formal model for a foraging swarm scenario to assess whether a swarm will indeed behave as required. Since swarm behaviors usually implement uncertainty and naturally depend on stochastic information, the authors have used a probabilistic model checker *i.e.* PRISM (introduced in [119]). Based on the work of [70], [71] proposed a formal model to assess the flexibility of a foraging system. The flexibility concept was redefined as the ability of a system to achieve the same collective behaviors in diverse environments without changing the behaviors of the individual agents. Similarly, the PRISM model checker was used and the model was validated by extensive simulations in ARGoS [120]. [72] applied formal verification using temporal logic [121] with the model checker NuSMV [122] on a particular swarm aggregation algorithm, namely Nembrini's alpha algorithm. The authors show how formal temporal analysis can help to refine and analyze such an algorithm. In prior work, [73] focused on the alpha algorithm using temporal logic as well, but the emphasis was on specification rather than verification. [123] used UPPAAL [74] to formally model and verify two behavior algorithms *i.e.* path planning and live human navigation through the robot swarm. However, they did not take into consideration the stochastic behavior of the swarm or the temporal specification of the system. As part of the ASCENS project, [124] studied a collective transport behavior using the formal language Klaim and related analysis tools. A particular attention has been dedicated to the stochastic aspect of a problem. However, their approach is scenario-specific and can involve only a limited number of robots.

Overall, we believe this is a first attempt at formally modelling the robustness of consensus-based behaviors from a general standpoint. We propose a methodology that can be applied to multiple algorithms and that could be used to assess, validate, and compare the robustness of swarm behaviors.

4.3 Statistical Model Checking

Developing robot swarm algorithms is usually a trial-and-error process that stops when the desired collective behavior is achieved. This method is time consuming and does not guarantee the detection of all unexpected behaviors since it does not cover all system outcomes.

To circumvent this problem, formal methods in general, and Model Checking (MC) in particular, are a potential solution. The essence of MC is to check the system against a property stated in a specification logic. If the property is not satisfied, the checker reports a counter example.

The main limitation of this technique is the state explosion problem [125] which reduces its applicability to simple systems. Symbolic MC [126, 127], partial ordering and abstractions [128] are useful alternatives to attenuate the problem and are at the core of many powerful model checkers such as NuSMV [122], SPIN [129], UPPAAL [130], and others.

For deeper analysis, MC was extended and adapted to the new needs of designers who need to i) quantify the impact of the uncertainty factor in many systems exhibiting stochastic behavior (Probabilistic Model Checkers such as PRISM [119]; and ii) add explicit temporal features (*i.e.* real time) in the specifications to estimate the time when some situation will arise (with Timed Model Checkers such as UPPAAL). Furthermore, both UPPAAL and PRISM introduced new techniques to study both aspects together, *i.e.* probabilistic timed systems. However, they still suffer from the state explosion problem due to the exhaustive exploration of the model state-space.

Statistical Model Checking SMC [131] is an alternative to the exhaustive MC based approaches. The core idea of SMC is based on sequential hypothesis testing [132] or Monte Carlo simulation [133]. First, the SMC conducts some simulations of the system, then it uses their outcomes to decide whether the system satisfies the property or not with some degree of confidence. SMC proposes a tradeoff between standard testing approaches and the exhaustive formal techniques. SMC was implemented in UPPAAL, and defeated PRISM on several case studies [134].

In the SMC UPPAAL tool, a system can be defined as a network of Priced Timed Automata (PTA for short) [134, 135]. A PTA is an automaton extended with clocks, clock rates, probabilistic delays and weights. Clocks measure the delay between different actions in the model, and they evolve synchronously and continuously with time. Clock rates quantify the clock evolution in the different states of the model. Such rates are specified with ordinary differential equations. For instance, when a clock rate is set to 0 ($X' == 0$), the clock X is stopped until the PTA process leaves the corresponding state. For a network of Priced Timed Automata (NPTA) [134], we can associate probabilistic delays with different states to synchronize the PTAs (called also processes). If the current state of a PTA admits a bounded invariant, then the probabilistic delay to leave the state is taken according to a uniform distribution within the bounded invariant interval. If the state does not have an invariant over time, then the probability of leaving the state is calculated according to an exponential distribution as follows:

$$\mathbf{Pr}(\text{leaving after } \mathbf{t}) = 1 - e^{-\lambda t}$$

where t is time and λ is a user-defined parameter. The PTA with the shortest probabilistic delay is selected to execute its upcoming transition whose guards are satisfied. If there are several PTAs whose delays are equal (they have the same λ), then one of them is chosen randomly according to a uniform distribution. Weights denote the likelihood of taking different transitions in the model.

The probability of a particular transition $\mathbf{Pr}_i$ is determined as a ratio of its weight w_i over the sum of weights of all transitions emanating from the same state:

$$\mathbf{Pr}_i = \frac{w_i}{\sum_{j=1}^{m}(w_j)}. \tag{4.1}$$

In a PTA, a state is defined by the current location and current values of all variables. Transitions are annotated with guards, synchronization signals, blocks of actions, and selections. A transition is enabled in a state if and only if the guard evaluates to True. In this case, the actions' block of the transition is executed automatically. The side effect of actions updates the state of the system. Transitions labelled with complementary synchronization signals $c!$ and $c?$ over a common channel c must synchronize. The update of the sender $c!$ is executed before the update of the receiver $c?$. A broadcast channel is a non-blocking channel where a sender can synchronize with an arbitrary number of receivers. The selections allow to randomly initialize some variable in a given range whenever a transition is executed.

To specify the system requirements, SMC UPPAAL uses weighted temporal properties expressed in $WMTL_{<=}$ (Weighted Metric Temporal Logic, [136]). This logic is defined by the grammar $\phi ::= ap|\neg\phi|\phi_1 \wedge \phi_2|O\phi|\phi_1 \sqcup_{x<=d} \phi_2$ where ap is an atomic proposition, $d \in \mathbb{N}$, x is a clock , $\neg$ refers to negation and and O is a next state operator. The logic formula $\phi_1 \sqcup_{x\leq d} \phi_2$ is satisfied if ϕ_1 is satisfied until ϕ_2 is satisfied, and that before the value of the clock x increases by more than d. Other commonly used operators in $WMTL_{<=}$ can be defined using the above grammar. As an example, $\Diamond_{x\leq d}\phi = True \sqcup_{x\leq d} \phi$ and $\Box_{x\leq d}\phi = \neg\Diamond_{x\leq d}\neg\phi$.

$WMTL_{<=}$ also introduces a new operator Pr and expresses four types of properties:

- Probability Estimation: this query computes the number of runs needed to produce an approximation interval $[p-\epsilon, p+\epsilon]$ with a confidence $(1-\alpha)$ where: $p = Pr[bound](\phi)$: ϕ is the expression to be evaluated and $bound$ defines how to bound the runs. There are three ways to bound them i) implicitly by time by specifying $<= M$ (where M is a positive integer), ii) explicitly by cost with $x <= M$ (where x is a specific clock), or iii) by number of discrete steps with $\# <= M$. α is called the confidence parameter and it measures the number of false positives and ϵ is an approximation parameter. ϵ and α are chosen by the user and the number of runs relies on the Chernoff-Hoeffding bound [137];

- Hypothesis Testing: this query compares the probability of a given property ϕ to a certain threshold $p \in [0,1]$ *i.e.* $Pr[bound](\phi) <= p$. SMC UPPAAL implements this query using Wald's sequential hypothesis testing [138];

- Probability Comparison: this query is an extension to the previous one, where it is possible to

compare the probability estimation of two different properties ϕ_1 and ϕ_2 *i.e.* $Pr[bound](\phi_1) <= Pr[bound](\phi_2)$. SMC UPPAAL exploits an extended Wald testing. More details can be found in [134];

- Value Estimation: this query estimates the expected minimum or maximum values of an expression that evaluates to a clock or an integer value. SMC UPPAAL implements this query using the following syntax:

  $E[bound, N](min(or\ max) : expr)$, where N is the number of runs, and $expr$ is the expression to evaluate.

SMC UPPAAL offers an additional operator "simulate" which allows to visualize the values of expressions (evaluating to integers or clocks) along simulated runs. This operator provides a better understanding of system behavior and allows more interesting properties to be asked to the model-checker.

4.4 Formal Modeling

Our paper focuses on consensus-based algorithms implemented for robotic swarms, and in particular we rely on virtual stigmergy [66] as the underlying communication mechanism. Our goal is to demonstrate how to analyze and validate consensus behaviors using SMC. We assess the resilience of these behaviors in the presence of communication loss and robot failures, based on the size of the swarm and its communication topology.This can be done by considering the stochastic aspects of the system using measurable properties to express system requirements. We believe SMC to be an appropriate tool for this purpose. We use the SMC UPPAAL tool[1] as UPPAAL is known to be a standard in the modeling and verification of real-time systems.

To model the consensus problem, we provide two strategies which are able to produce the same statistical results. However, they are complementary in terms of advantages and weaknesses : i) **Agent-based model**: we represent a swarm of n robots by a network of n identical PTAs. This model allows a better understanding of the agents' interaction using broadcasting channels. However, it is less scalable due to the state explosion problem. More on this in Section 4.6. ii) **Swarm-based model**: we specify a swarm network with one automaton. The model abstracts the agents broadcasting and focuses on the stigmergy status evolution. This model is less expressive, but exhibits better scalability (see Section 4.6).

Although our models are generic and can be applied to a wide range of scenarios, we chose to validate it on a scenario that has been already designed, implemented, and tested in [66]: the

[1] http://people.cs.aau.dk/∼adavid/smc/

election of a leader. Moreover, we show how to extend the model to a more complex scenario (task allocation) with minor effort.

4.4.1 Consensus Strategy Specification

In this section, we formally specify the consensus strategy used to implement the behaviors of interest.

We define *consensus* as the agreement of all robots in a swarm over a common value representing their state or knowledge of the environment. We call *barrier* the implementation of the consensus among the robots in the swarm. A barrier practically allows the swarm to wait for agent consensus under certain conditions before moving to the next behavior. A detailed implementation of the barrier concept is provided in [139].

Let us represent a swarm with a labelled graph defined as a tuple $G = (\mathcal{R}, \mathcal{VS}, \Lambda)$, where:

- $\mathcal{R} = \{r_0, r_1, \ldots, r_{n-1}\}$ is a set of robots identifiers (n is the number of robots in the swarm),
- A virtual stigmergy $\mathcal{VS} = (\mathcal{E}, \psi, \omega)$ is a shared table among all the robots in the swarm where:
  - $\mathcal{E} =< e_0, e_1, \ldots, e_{z-1} >$ is a dynamic set of entries where an entry $e = (k, v, ts, r)$ consists of a record containing a key k, a value v, a timestamp ts (which introduces a temporal ordering on the entry updates), and a robot identifier $r \in \mathcal{R}$ that records the robot originating the entry e.
  - ψ is a partial function $\mathbb{N} \times \mathbb{R} \times \mathcal{R} \mapsto \mathcal{E}$ that given a tuple (k, v) creates a new entry $e = (k, v, 1, r)$ when e does not exist in the table.
  - ω is a partial function $\mathcal{E} \times \mathbb{N} \mapsto \mathbb{R}$, that given a set of entries $\mathcal{E}$ and an integer k returns the value v matching k in $\mathcal{E}$ or 0 if no matching tuple exists.
- Λ is a function $\mathcal{R} \times \mathcal{R} \mapsto \{0, 1\}$, which associates with each couple of robots 1 if they are neighbors and 0 otherwise.

For a typical task execution scenario, we can define $\mathcal{T} = \{t_0, t_1, \ldots, t_{l-1}\}$ as a set of tasks where l is the number of tasks. A barrier $\mathcal{B}$ is associated with each task t_i in $\mathcal{T}$ and can be formally defined as a binary function that returns true if consensus is reached.

We use the virtual stigmergy $\mathcal{VS}$ to implement a barrier mechanism that works as follows: whenever a robot r_i needs to update an entry e, the change is made first in its local table. If the entry does not exist, robot r_i calls the ψ function to create a new entry $(k, v, 1, r_i)$; if the entry exists

its value is updated and its timestamp is increased. Subsequently, the robot broadcasts a packet containing the updated entry to its neighbors. Whenever a robot needs to read a tuple in its local copy of $\mathcal{E}$, it uses the ω function. If the tuple is unknown, a nil value is returned. Conversely, if the tuple exits, the robot broadcasts a packet to its neighbors with the requested entry to ask whether it is up-to-date. This broadcast mechanism allows robots to recover data integrity in case of random packet drop or connectivity loss due to some robots' failure. The reader can refer to [66] for more details.

Before moving to the next task, the swarm waits until all members have reached the barrier for the current task. In other words, the barrier halts the robots' task until they reach global consensus. We use a timeout function to avoid deadlock. It is worth noting that the logic of the barrier function is scenario-dependent. For example, let us consider our first scenario where the robots in the swarm must agree on the highest robot identifier to be elected as a leader. In this case, $\mathcal{B}$ would return true if and only if the value associated with the key of the requested information (the ID of the leader) is the same for all robots. For our second scenario bidding-based task allocation, $\mathcal{B}$ assumes that all robots in the swarm must agree on the assignee of the current task.

We chose these two scenarios for several practical reasons: 1) these two behaviors are among the most common in swarm robotics; 2) their simplicity allows us to abstract task complexity and focus on the global behavior of the systems; and 3) they are usually part of many more complex behaviors: many scenarios require robots to allocate tasks and/or to elect a leader.

We consider three classes of swarm topologies (see [66]):

- **Cluster topology**: robots are placed randomly in a high-density cluster. Robots in the middle have typically more neighbors than those on the edges. This topology implements a very optimistic case where packet drops or robot failures are unlikely to increase the convergence delay, the time needed to reach consensus;
- **Line topology**: robots form a straight line. Each robot has exactly two neighbors, except the first and the last one who have only one neighbor. This topology is extremely sensitive to robots' failures and packet drop probability.
- **Scale free topology**: it is a small-world network where robots form multiple clusters connected by lines of robots [140]. This topology represents an intermediate case where the number of connections in the swarm is in between the two previous extreme cases.

These topologies can be classified by their connectivity rate. This later can be computed as:

$$\textbf{connectivity\_rate} = \frac{\text{nlinks}}{n(n-1)} \tag{4.2}$$

where nlinks is the number of active connections in the swarm, n is the number of robots and $n(n-1)$ is the total number of possible connections if the agents are connected in a peer-to-peer fashion. According to Equation 4.2, the cluster, line, and scale free topologies have a high, low, and medium connectivity rate respectively.

4.4.2 Agent-Based Specification

We represent a swarm of n robots by a network of n identical PTAs, each with a unique robot ID (denoted RID in the model). Fig. 4.1 represents a PTA-based model for one robot in the swarm. As explained above, PTAs can exchange data by broadcasting and receiving packets simultaneously via the broadcast channel *send*. A packet represents an entry (k, v, ts, r_i) in $\mathcal{E}$. We used the clock variable *timecost* to record the time needed to reach consensus. On the *END* state, we imposed a null clock rate over *timecost* to stop the clock. To synchronize the PTAs, we set the exponential rate λ to 1 on the initial states *START* of all the model's PTAs. Consequently, at each time step, a PTA is chosen randomly, according to a uniform distribution, to execute its upcoming action.

Initially, the model is configured by means of a generator automaton which is described in Fig. 4.2. The generator first uses the model function *Configure_Networ-k()* to configure the swarm according to two model parameters *i.e.* $size$ and $topology$.

The generator automaton uses then the model function *Set_Barrier(task)* to configure the initial state of the robots. For the leader election scenario, each robot initially creates an entry in its local VS table where the entry key and value are set to the task identifier and the robot identifier, respectively. Finally, the generator starts the election process by setting the Boolean variable *start_swarm* to true.

According to the $\mathcal{VS}$ protocol (Fig. 4.1), each robot in working order (*Is_Failed()* returns false) starts broadcasting its state. Neighbors who are within the communication range of the sender and for whom *Is_Connected()* returns true, receive the message with a certain probability $1 - \boldsymbol{P}$ (we used the PTA weights feature where $\boldsymbol{P}$ is the Probability of Packet Loss) and update their tables using the model function *Update()* . For the leader election scenario, the *Update()* function returns the remote value if and only if the sender value is greater than the local one, or the two values are equal and the robot identifier of the remote entry is greater than the local one. Otherwise, the robot keeps its state unchanged.

It is worth noting that the timestamp attribute ts is not used for this scenario because each robot changes its value only according to what it receives from its neighbors. At each time step, each robot broadcasts its state and asks the neighbors whether the data is up-to-date until it reaches consensus (*i.e.* the *Converge (task)* function returns true) or the task elapsed time (*i.e.* model

variable *timecost*) exceeds a certain threshold (*i.e.* model variable *timeout*). If either of those two cases is true, the model reaches the *END* state and declares the task either Done or Failed.

In the model, each robot may fail with a certain probability ***Pf*** (see Equation 4.1). This probability is used to study the robustness of the swarm to faults. We use the function *Update_Network(r)* to delete a failed robot r and its connections from the swarm topology.

4.4.3 Swarm-Based Specification

To make the model more robust to the explosion state space problem, we abstract the agents broadcasting and focus on the stigmergy status evolution. The new specification is depicted in Fig. 4.3. More precisely, we limit the agents' interactions by creating only two automata to mimic the swarm behavior. The agent automaton depicted in Fig. 4.3(b), statically records which agents will receive the information from a sender, according to the model parameter ***P*** (probability of packet loss). Hence, we create for each broadcast operation a static list of receivers using an attribute in the robots' data structure *i.e.* ***recv*** is set to 1 if the robot receives the information and -1 otherwise. The swarm automaton depicted in Fig. 4.3(a) selects a sender uniformly from the set of non failed robots (*Is_Failed(sender)* returns false). After broadcasting the information, the sender resets the receivers list by putting the robots' attribute ***recv*** to 0 using *Reset_recv()*. In the *record* state, the clock is stopped (timecost'==0) to avoid biasing the time to consensus by the static recording operation. Therefore, all robots receive the information sent simultaneously and instantly, as with our first model. The *Fail* flag is used to make sure that the model returns to the *START* state when the sender passes through a fail state. It is worth mentioning that like the first model, the generator automaton 4.2 is used to initialize the network topology and the stigmergy table for each robot. This abstraction shifts the model complexity from n dependent automata to 2 independent automata (the static agent automaton and the swarm automaton). This abstraction decreases dramatically the execution interleavings during the verification process. Consequently, the state space can be greatly reduced and the combinatorial explosion problem can be greatly limited.

It is worth to note that the two specifications, we have presented, have the same dynamics and are able to produce the same results. The agent based model highlights the interactions between the robots, which allows a better understanding of the problem, whereas the swarm-based model abstracts the physical interaction and focuses on the information evolution for better scalability. Both models work by uniformly choosing, at every step, a sender from the set of the available robots. The information sent by the selected sender is received simultaneously and instantly by all the connected robots, filtered by the probability of packet loss.

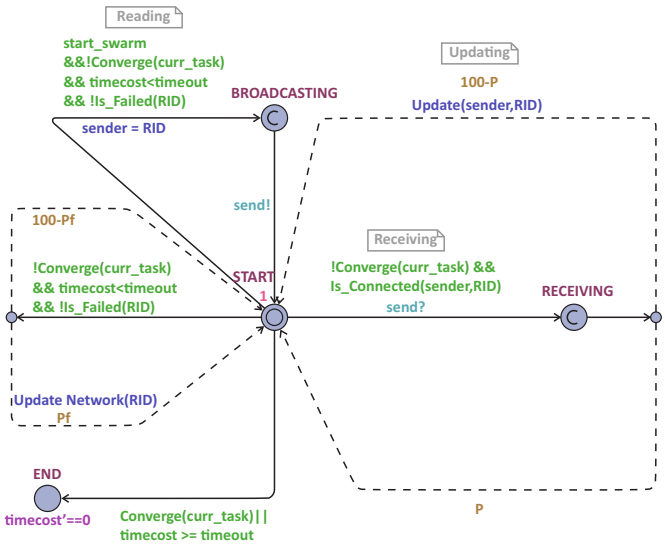


Figure 4.1 Agent-based specification

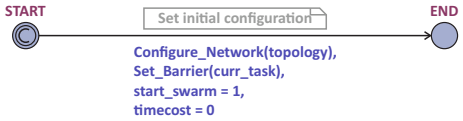


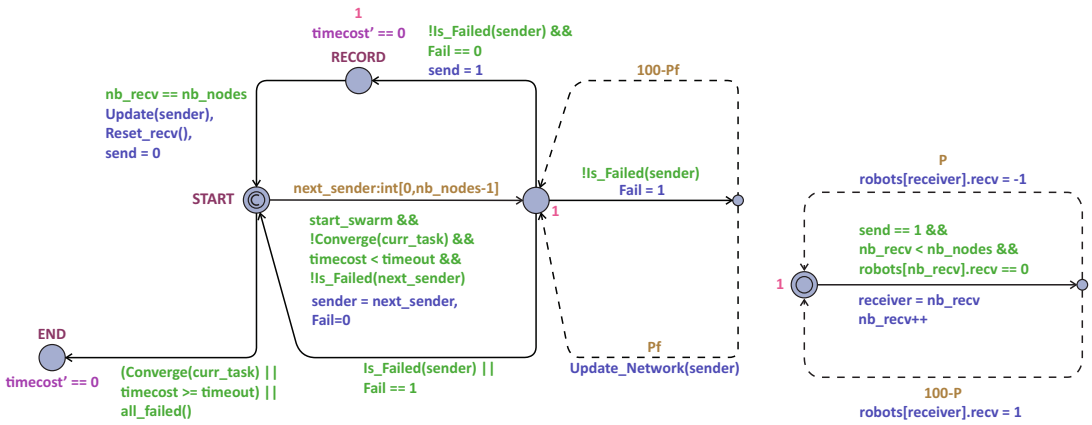
Figure 4.2 Generator automaton

4.4.4 Model Extension for a Bidding-Based Scenario

In this section, we show how to extend our formal model to study more complex scenarios with very little effort. A common scenario for a robot team is to execute a given queue of tasks evolving throughout a mission. In such scenario, the swarm should partition the available tasks and agree on a set of allocations. There are multiple task allocation approaches for autonomous swarming behaviors [116]. One popular and powerful technique is the auction based allocation, which consists of selling a set of tasks to bidders through four phases: task advertising, bidding, solving the auction, task assignment [141].

Auction algorithms differ based on various criteria such as the number of rounds needed to sell a certain number of tasks, how many tasks each robot can bid for, and how many tasks are sold in each round. In this paper, we model the case of a sequential auction where the number of rounds corresponds to the number of tasks *i.e.* only one task is sold in each round. Each robot can bid for each advertised task using a specific value function depending on the problem being solved. The task is then assigned to the auction winner based on the consensus principles (see Section 4.4.1).

For our bidding based scenario, for the sake of simplicity we consider a semi-distributed system in which one robot is elected to be the auctioneer, for example, using the previous leader



(a) Swarm's network automaton (b) Static agent automaton

Figure 4.3 Swarm-based model.

election scenario (Fig. 4.4). Using the virtual stigmergy, it is possible to implement a fully distributed system, but that would require a description of its inner mechanisms that are out of the scope of this paper. Nevertheless, the same methodology applies. For abstraction, we assume that the advertisement to be completed and that tasks are defined uniquely by their identifier. As by [141], we consider that the bids are continuous random variables, that is to say the winner is chosen non-deterministically from the set of robots. This guarantees that each possible ordering of robot-task valuations is equally likely. This can be easily modeled using the UPPAAL select tool $winner : int[0, n\_robots - 1]$. After solving the auction by selecting an appropriate winner for whom *Win_Bid () returns true*, the auctioneer updates its table ($e.v = winner$) using the *Update_Inner_State (winner)* function and locks the auction process by lowering the *start_auction* flag. Consequently, the auctioneer broadcasts its decision to its neighbors via the *send* channel. The information then goes through the virtual stigmergy process until a consensus is met.

Since our PTA models depicted in Fig. 4.1 and Fig. 4.3 are generic, we use them for the bidding based scenario as well. We modify the *Update()* function so that the receiver updates its state only if its timestamp is smaller than the sender's. Whether or not the swarm reaches consensus before the $timeout$, the function *Move_to_Next()* is set to true and the model calls *Set_Task_status()* to set the task status to i) "allocated" if *Converge()* returns true, and the winner adds the task to its task queue; or ii) "non-allocated", otherwise. Subsequently, the auctioneer sets the barrier of the next task using the *Set_Barrier(curr_task)* function and raises the *start_auction* flag. When the model has allocated all the given tasks, the *End_Auction()* function returns true and brings the model to the *END* state. The generator automaton (Fig. 4.2) is used to configure the network and initially set the barrier for the first task.

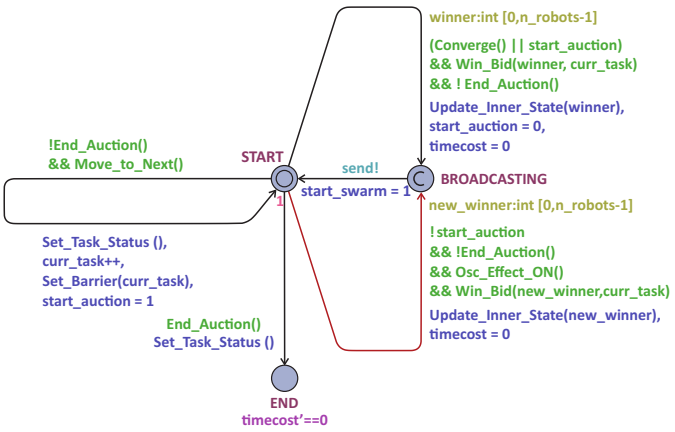


Figure 4.4 Auctioneer Automaton

4.5 Statistical Verification

As discussed in section 4.3, the core engine of SMC UPPAAL offers a set of queries that allow to verify a network of PTAs against a given system specification.

To study the stochastic behavior of both scenarios, we use the following methodology:

- We first ensure the correctness of the model enforcing a set of qualitative properties implemented in the classical model checker engine of UPPAAL. These properties are used to verify the proper functioning of the model and the consensus system. Here, we chose the swarm size in the set $[5, 10, 20]$, the topology in [cluster, line, scale free], ***P*** in $[0\%, 25\%, 50\%, 75\%, 95\%]$ and ***Pf*** in $[0\%, 2\%]$.

- For the leader election scenario, we used the equation 4.7 which computes the probability the consensus is met for a specific task (*Converge(task)* returns true). The equation 4.7 is checked for all swarm topologies (swarm size = 10) to assess i) the impact of packet loss on the probability of reaching consensus and on the time needed to complete the election task (***P*** in $[0\%..95\%]$), and ii) the robustness of the swarm behavior to robots' failure (***Pf*** in $[0\%..100\%[$).

- For the bidding based scenario, we assess i) the impact of packet loss and robots' failures on the number of allocated tasks, and ii) the impact of the oscillation effect (i.e. information stability) on the time needed to complete the tasks. Here, we check all swarm topologies where the swarm size is in the set $[5, 10]$, the number of tasks is in the set $[10, 20]$ and ***P*** and ***Pf*** are as defined in the previous step.

The confidence parameter α and the approximation parameter ϵ are both set to 0.05 (details on

these two parameters are in Section 4.3). All measurements are given with a confidence rate (1-α) of 95%.

4.5.1 Qualitative Verification

We first check that our model is free of deadlock states (*i.e.* deadlock is a state where there are no outgoing action transitions neither from the state itself or any of its successors); that is to say, the model should exhibit a deadlock if and only if a decision is made (whether consensus is reached or not). This property is expressed using two queries:

$$\Box\ (deadlock\ imply\ (task.isdone\ ||\ task.isfailed)) \tag{4.3}$$

$$\Box\ ((task.isdone\ ||\ task.isfailed)\ imply\ deadlock) \tag{4.4}$$

These two queries are satisfied for all the tested model configurations. Moreover, we investigated the ability of the swarm to reach consensus under some configurations using the reachability property:

$$\diamondsuit\ (Converge(task).istrue) \tag{4.5}$$

This property evaluates to true if the system can reach a state where the swarm successfully achieves the task of electing the robot with the highest identifier. This property was satisfied for some of the defined model configurations.

Using the property:

$$\diamondsuit\ (Converge(task).isfalse), \tag{4.6}$$

we have noted that the swarm may not converge under certain conditions, such as high packet loss or with a line topology where robot failure probability is not nil. In such cases, the system may exhibit additional delays and therefore exceed the *timeout* threshold associated with the given task.

4.5.2 Statistical Verification of Electing a Leader

Impact of packet loss: In the leader election scenario, the swarm keeps exchanging data until it meets a convergence state or the time assigned to the task runs out. This latter condition is controlled by a user-defined parameter called *timeout*. To properly set this parameter, we used the exceedance probability which is plotted as $1 - cumulative\ probability$ of reaching consensus over the range of *timeout* values. The exceedance probability is used as a threshold (10^{-15}) to determine the maximum time needed by the swarm to reach consensus. The data are collected for a swarm of 10 robots using property 4.7 with different *timeout* values. As shown in Fig. 4.5, in the most pessimistic case where the communication quality is very poor *i.e.* $P = 95\%$, the *timeout* parameter

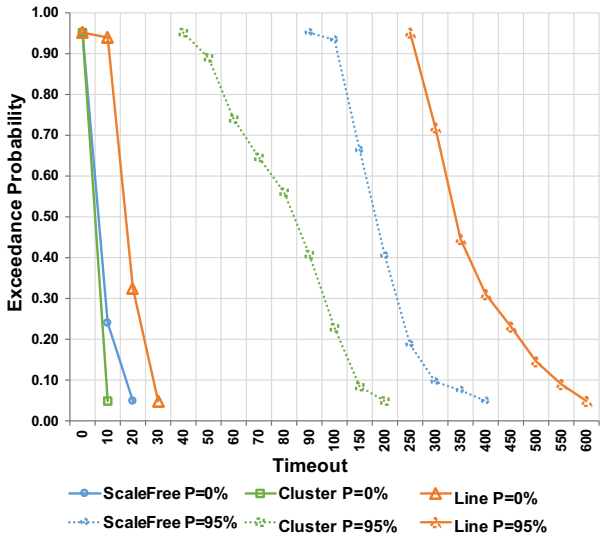


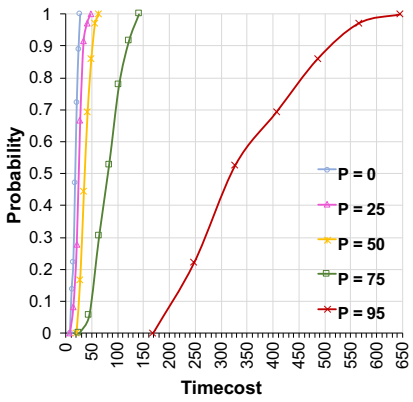
Figure 4.5 Exceedance probability to reach consensus for different *timeout* values

should be set to at least 650, 450 or 250 time steps to ensure convergence for a line, scale free or cluster topology, respectively. In the case where the *timeout* is set to smaller values, the probability of reaching consensus will decrease (according to Fig. 4.5) given the extra delays for high packet loss probabilities.

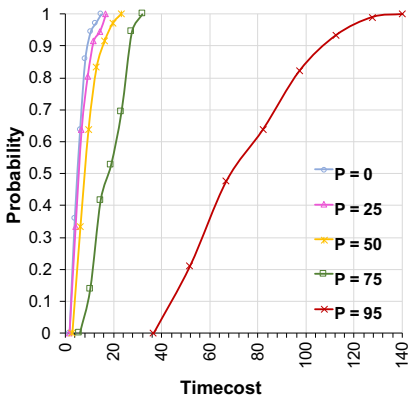
$$Pr[<= bound](\Diamond Converge(task).istrue) \tag{4.7}$$

To better understand the impact of the communication quality on the convergence time (*timecost*), we performed a deeper analysis using property 4.7, computing the probability of reaching consensus for different packet loss probabilities. The bound value (see Section 4.3) is set to 1000. This value is chosen empirically after a set of trials (see results on Fig. 4.5) showing that in the most pessimistic case the system does not exceed 650 time steps to decide about the election task. To give the model higher degree of freedom while checking the property, we considered an additional margin of around 45% of the worst case time, or 1000 time steps. The equation 4.7 also returns the calculated probability, and when the robots reach consensus.

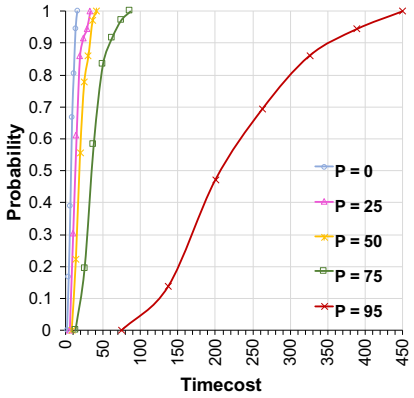
Fig. 4.6 plots the cumulative probability of convergence in terms of *timecost* for different packet loss values for a swarm of 10 robots in a line (Fig. 4.6(a)), cluster (Fig.4.6(b)) and scale free (Fig.4.6(c)) topologies. Fig. 4.6 shows that within the worst case *timeout* (650 time steps in our case), the swarm is able to complete the election task with a probability of almost 100%. The line topology is the most sensitive to the communication quality– the higher the packet drop probability,



(a) Cumulative probability to reach consensus for a line of 10 robots



(b) Cumulative probability to reach consensus for a cluster of 10 robots



(c) Cumulative probability to reach consensus for a scale free topology of 10 robots

Figure 4.6 Cumulative probability to reach consensus in terms of time cost for different packet loss probabilities

the longest is the convergence delay. The cluster topology is a very optimistic case, in which communication problems are less likely to increase the convergence time. The scale free topology falls in between the two previous cases. For example, when $P = 75\%$, the *timecost* of electing a leader for a line topology is about 150 time steps whereas a cluster topology takes only around 30 time steps to successfully achieve the task. The scale free topology, in this case, takes around 90 time steps.

Note that in the following we use the simulation term to refer to model sampling, and it should not be confused with robot simulations.

One of the most powerful properties of SMC UPPAAL is the maximum/minimum value estimation:

$$E[<= bound; nb\_simulations](max : timecost) \tag{4.8}$$

This property allows us to estimate the maximum $timecost$ the swarm may exhibit to elect a leader. We set the $bound$ value to 1000. As explained above, this value is chosen empirically from Fig. 4.5 which shows that in the most pessimistic case the system does not exceed 650 time steps to decide about the election task. An additional margin of around 45% of the worst case time is added to give the model higher degree of freedom when checking the property. We chose to adjust $nb\_simulations$ to 5000 because this value gives a good tradeoff between accurate Monte Carlo modelling and verification time (the higher the number of simulations the more accurate the results and the longest is the verification time).

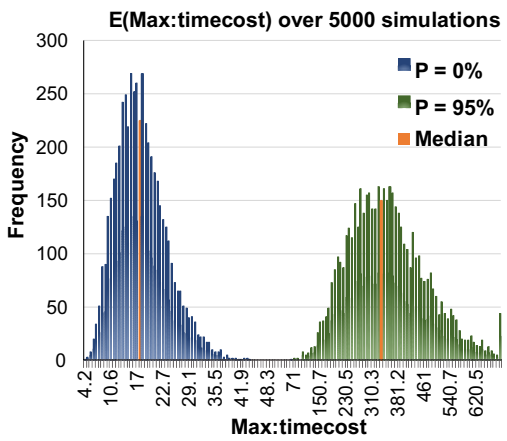
Fig. 4.7 shows the maximum *timecost* for a line 4.7(a), cluster 4.7(b) and scale free 4.7(c) 10-robot-topologies. The packet loss probability ***P*** is set to 0% and 95%. For a line, the median of the maximum *timecost* (over 5000 simulations) is 18.13±0.169 time steps for $P = 0\%$ and 356.83 ± 3.248 time steps for $P = 95\%$. As expected, the line topology exhibits important delays with higher packet loss probabilities. These delays are less important for the scale free topology where the median of the *timecost* is 8.00±0.109 for $P = 0\%$ and 198.28±2.223 for $P = 95\%$. The smallest delays are recorded for the cluster topology (5.39±0.071 for $P = 0\%$ and 87.72±0.961 for $P = 95\%$).

Impact of robot failures: To study the fault-tolerance of the swarm, we assessed property 4.7 while varying the robot failure probability ***Pf***. Fig. 4.8 depicts the probability of reaching consensus with a 10-robot swarm when $P = 30\%$ in terms of failure probabilities for scale free, cluster and line topologies. The figure shows that the scale free topology is more tolerant to failures than the line topology. For the scale free topology, the probability of successfully electing a leader tends gradually to 0 when ***Pf*** increases and it vanishes starting from a failure probability of 60%. Whereas, for the line topology, the probability of convergence is already nil for much smaller ***Pf*** values: there is no chance to reach consensus when ***Pf*** exceeds 8%.

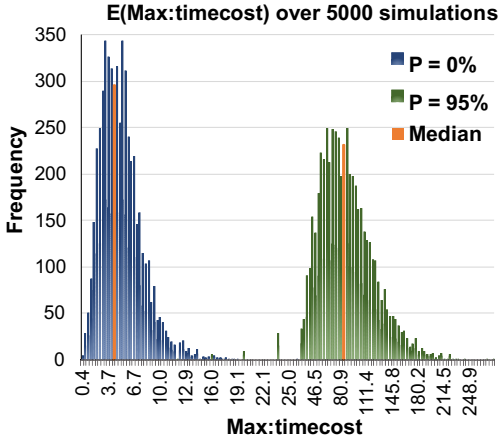
The cluster topology (in green) is, as expected, the most tolerant to faults and connectivity problems.

To better understand the behavior of the scale free topology (our middle-ground case), we plot on Fig. 4.8 the probability of losing all the robots in the swarm using:

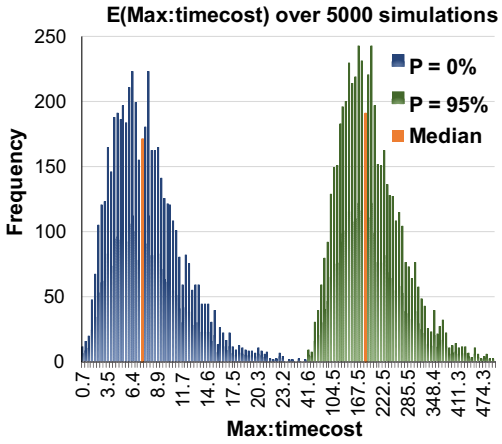
$$Pr[<= bound]\ (\diamondsuit\ forall(i : int\ [0, nb\_robots - 1]) \\ robots[i].isfailed) \tag{4.9}$$



(a) Maximum timecost distribution for a line of 10 robots with $P = 0\%$ and $P = 95\%$



(b) Maximum timecost distribution for a cluster of 10 robots with $P = 0\%$ and $P = 95\%$



(c) Maximum timecost distribution for a scale free topology of 10 robots with $P = 0\%$ and $P = 95\%$

Figure 4.7 Maximum timecost distribution

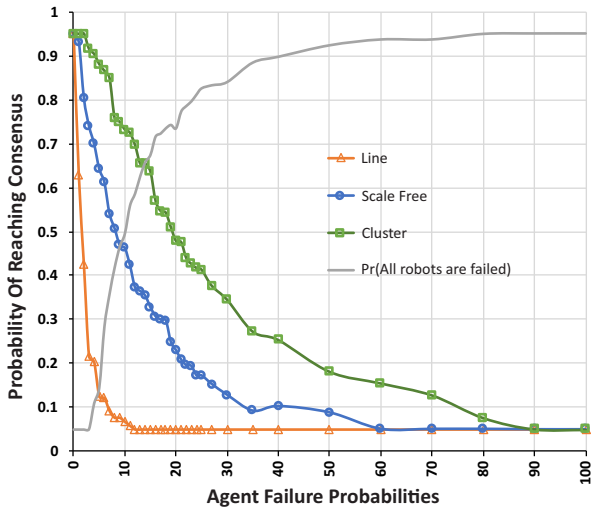


Figure 4.8 Impact of robots failing on the convergence for different swarm topologies $P = 30\%$

The probability of reaching consensus for the scale free topology is inversely proportional to the probability of losing all the robots. With higher failure probability, the swarm is more likely to lose all its robots and consequently, no robot would be available for election.

4.5.3 Statistical Verification of the Bidding-based Task Allocation

Impact of packet loss: One of the most important aspects we can study for this scenario, is how the communication quality affects the performance of the auction algorithm presented in Section 4.4.4; in particular, the expected number of allocated tasks, the expected number of tasks won by a robot, and the probability a robot is not assigned any tasks.

$$\textbf{simulate } 2000\ [<= 1000]\ \{auctioneer.allocated\_tasks\} \tag{4.10}$$

The expected number of allocated tasks within 1000 time steps is calculated using property 4.10. For different ***P*** values, this property runs 2000 model simulations and records, at each time step, the number of tasks that are assigned to the swarm robots. Fig. 4.9 displays the average of the expected number of allocated tasks, showing that, in the case of a perfect communication ($P = 0\%$), a 10-robot scale free swarm is able to successfully allocate 20 tasks in less than 100 time steps. For higher packet loss probabilities, the swarm needs much more time to allocate the 20 tasks. For instance, for $P = 75\%$, the tasks are allocated within around 700 time steps. When the probability of the packet loss is high, the swarm needs more than 1000 time steps to allocate all the tasks, for example only 4 tasks are successfully allocated with $P = 95\%$.

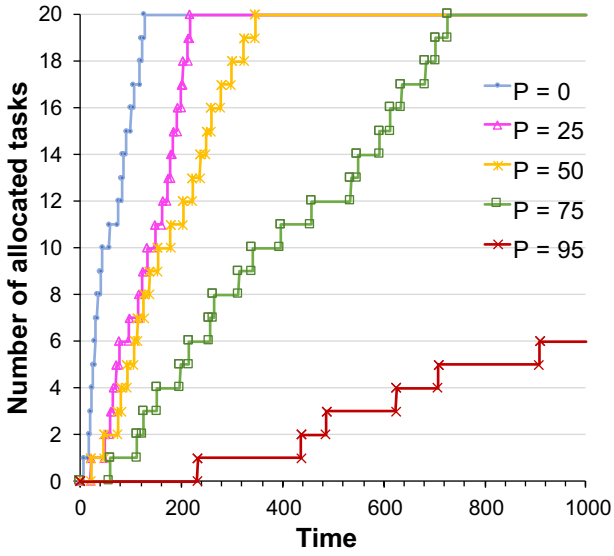


Figure 4.9 Number of allocated tasks in a scale free topology of 10 robots with 20 tasks for different packet loss probabilities in terms of time

Using property 4.10, we also run 2000 simulations of a 5-robot scale free swarm (with 10 tasks) to compute the number of tasks a robot may win over 1000 time steps (see Fig. 4.10). For $P = 0\%$, the robot's *won-tasks* variable is normally distributed ($\mathcal{N}(Median = 2,\ \sigma^2 = 5.63)$). Given this distribution, the probability that a robot is assigned exactly zero tasks is about 7%. On average, a robot can win 2 tasks out of 10, and it has a very little chance to win more than 6 tasks. For $P = 95\%$, Fig. 4.10 shows that the robot's *won-tasks* variable is represented by $\mathcal{N}(Median = 0,\ \sigma^2 = 1.71)$; in this case, a robot has a much higher chance of completing no tasks, and has almost no chance of having more than 3 tasks in the best case.

Impact of robot failures: To study the impact of the robot failures, we use property 4.10 while varying the ***Pf*** parameter. We have calculated the expected number of allocated tasks for different swarm topologies against different failure probabilities. The packet loss probability is set to 30%. Fig. 4.11 shows that a 5-robot swarm exhibits the same tolerance to failures for both cluster and scale free topologies. When the probability of a robot failure increases, the number of allocated tasks decreases for mainly two reasons i) there are not enough resources to assign tasks and ii) some robots may be isolated from the swarm and consequently, they cannot achieve consensus. Moreover, Fig. 4.11 shows the sensitivity of the line topology to failures. When the probability of failure exceeds 10% the swarm won't be able to allocate any of the tasks.

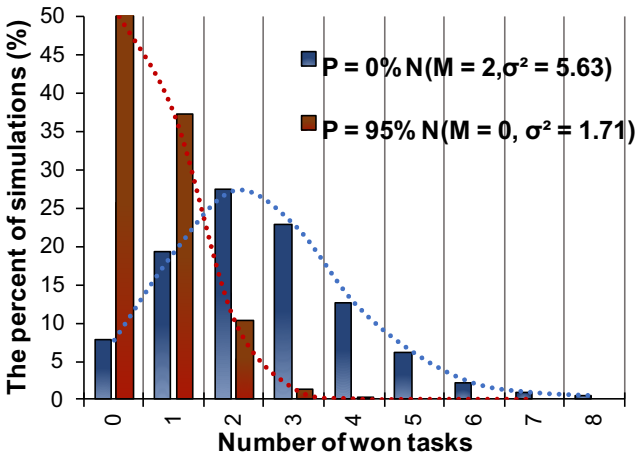


Figure 4.10 Number of tasks won by a robot over 1000 time steps

Impact of the oscillation effect: One of the important problems of a consensus based communication is information stability. For the virtual stigmergy system, oscillations occur when the information being shared often changes in a shorter period than the time needed to reach consensus.

To study this phenomenon, we force the Auctioneer automaton to change its mind as many times as possible (see Fig. 4.4 *Osc_Effect_ON* variable is set to 1) about the chosen winner before the swarm agrees on the previous winner. Fig. 4.12 plots the probability of allocating all tasks with and without oscillations for the three topologies (5 robots, 10 tasks, $P = 50\%$). Results show that even with oscillations, the swarm under any topology is able to allocate all tasks with a probability of almost 100%. The line topology seems to be the most sensitive to oscillations since it exhibits much longer delay to agree on allocations when the oscillation effect is present.

4.6 Model Performance

We provide two congruent SMC-based abstractions of a consensus behavior. The use of these mathematical models allows to virtually analyze and check the system and reduces the need for experimentation, which can be costly and time-consuming. Instead, the proposed models are used to simulate the swarm behaviors of interest and highlight the impact of some parameters on the robustness and the resilience of the system cost effectively. The relationship of equivalence that correlates the formal abstractions and the actual physical system is direct in the first model, as each robot is modeled as a sending/receiving entity. In contrast, the second model abstracts this relationship by encapsulating the broadcasting interactions, but keeping the same semantics. In this section, we discuss this relationship by comparing our SMC models with a physics-based simulator and real world experiments, as well as evaluating the approach in terms of verification

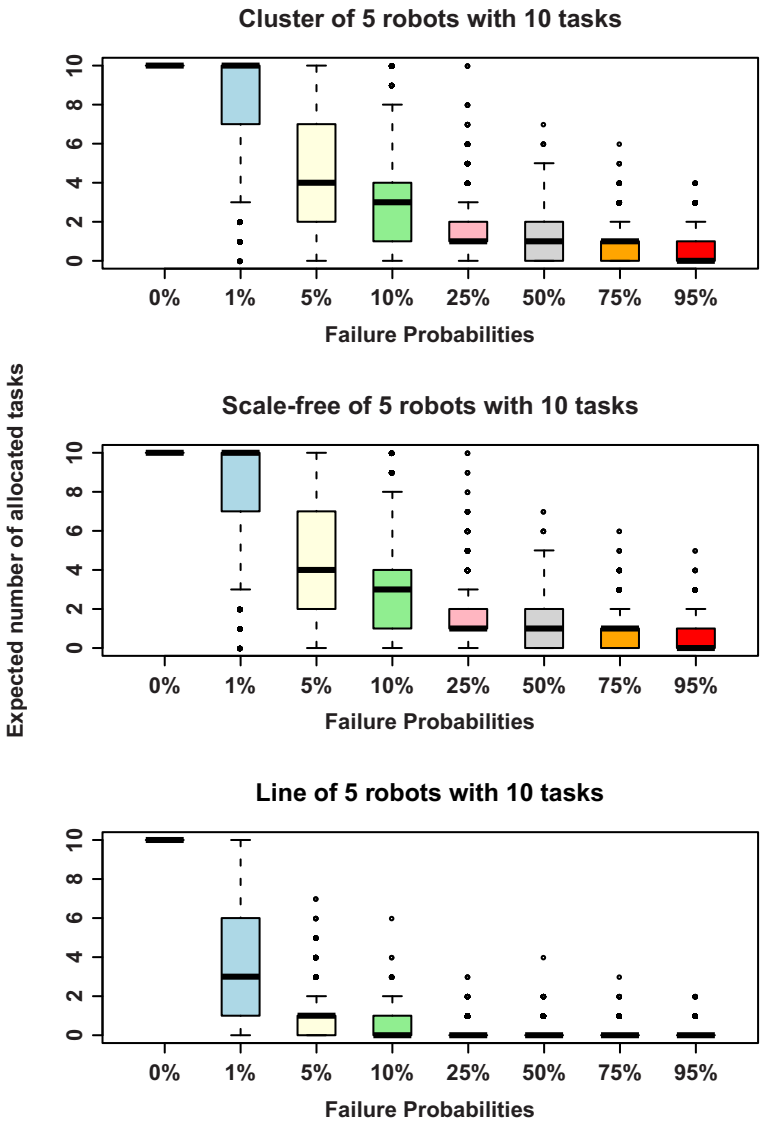
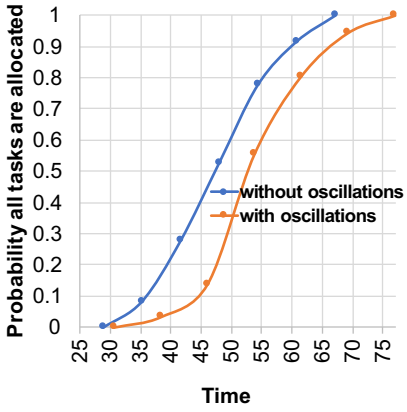
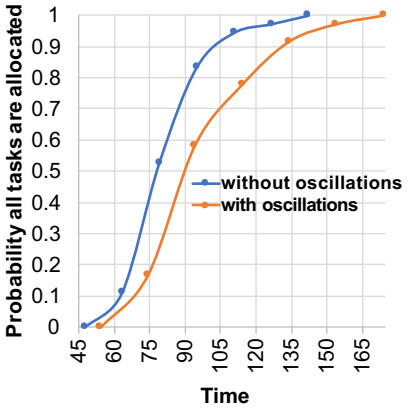


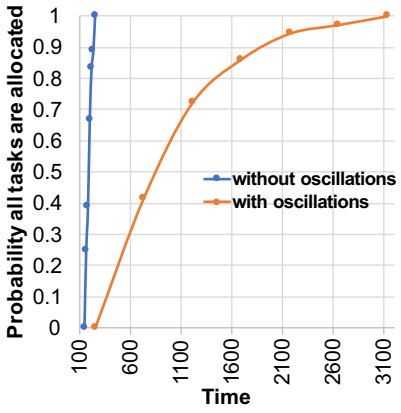
Figure 4.11 Impact of the robot failure probability on the number of allocated tasks for different topologies



(a) Cluster topology, 5 robots, 10 tasks and $P = 50\%$



(b) Scale Free topology, 5 robots, 10 tasks and $P = 50\%$



(c) Line topology, 5 robots, 10 tasks and $P = 50\%$

Figure 4.12 The impact of the oscillation effect on the probability that all tasks are successfully allocated

time and memory usage.

We implemented the leader election scenario in Buzz, a programming language for robot swarms [142].

- We used the ARGoS multi-robot simulator [120]: we have conducted 500 simulations on a 5-robot cluster with different packet loss probabilities, that is 100 simulations for each $\boldsymbol{P}$ value selected from $[0\%, 25\%, 50\%, 75\%, 95\%]$. For each case, we recorded the number of time steps needed to reach consensus;
- We implemented the scenario on 5 Khepera IV [2] robots connected by a standard $2.4GHz$

[2]https://www.k-team.com/mobile-robotics-products/khepera-iv/introduction

wireless network. Fig. 4.13 shows the 5 robots before reaching consensus (most of them display red LEDs, see Fig. 4.13(a)) and after reaching consensus (all LEDs are green, see Fig. 4.13(b)). Moreover, we controlled the packet loss probability by using a software hub Blabbermouth[3].

For the sake of brevity, we only present the results for the cluster topology. Results for the line and scale free topologies can be found in Appendix A. The consensus time ($timecost$) was recorded in ARGoS (time steps), Kheperas (ms) and our formal model (UPPAAL time step). In Fig.4.14, the recorded time is plotted against different packet loss probabilities. For consistency, it is normalized to properly compare the shapes of the three curves. Overall, the three curves have similar shapes, confirming that the model is representative of the real robot behavior. For small packet drop probabilities (less than 50%), ARGoS is the most pessimistic. However, it is worth noting that for small packet loss probabilities, the time needed to reach consensus for 5 robots is too short in the three cases. Therefore, it is difficult to discern the difference or to draw a conclusion. For higher packet loss probabilities, Fig. 4.14 shows that the SMC model is slightly pessimistic comparing with ARGoS and real world experiments, which is acceptable for conservative estimates. Although the SMC model may ignore some dynamic aspects of the actual system, and thus leading to some discrepancy in the curves, the overall behavior is well represented.

Furthermore, we study the performance of our model by comparing its two version (*i.e.* agent-based and swarm-based versions) in terms of computing time and the memory usage. The measurements were performed using the 4.0.19 UPPAAL version installed on Ubuntu (Core i7 32GB RAM). We present experiments performed on property 4.7, arguable the most important property in the model *i.e.* the probability of convergence.

We analyze the ability of the models to handle swarm of increasing sizes. Figures 4.15(a) and 4.15(b) depict the agent-based model performance in terms of the number of agents and the connectivity rate. These two figures show that the agent-based model can check a swarm composed of up to 400 agents fully connected (connectivity rate 100%). The model exhibits an increasing computing time as the number of agents and the connectivity rate in the swarm increase. The memory usage increases exponentially with size whereas it seems that it is not significantly hindered by the connectivity rate increasing. In a fully connected swarm of 100 robots, the computing time is around 600 seconds, with a memory usage rate close to 300 MB. For 400 fully-connected robots, the computing time reaches 9h and the model consumes as about 15 GB of memory. The model can also verify swarms of 500 agents with a connectivity rate of 60% in around 15h using about 27 GB of memory. These statistics exhibit the model's ability to handle relatively large networks with high connectivity rates.

[3]https://github.com/MISTLab/blabbermouth

(a) Before Consensus (Red LEDs) (b) After Consensus (Green LEDs)

Figure 4.13 A group of 5 Khepera robots electing a leader using virtual stigmergy.

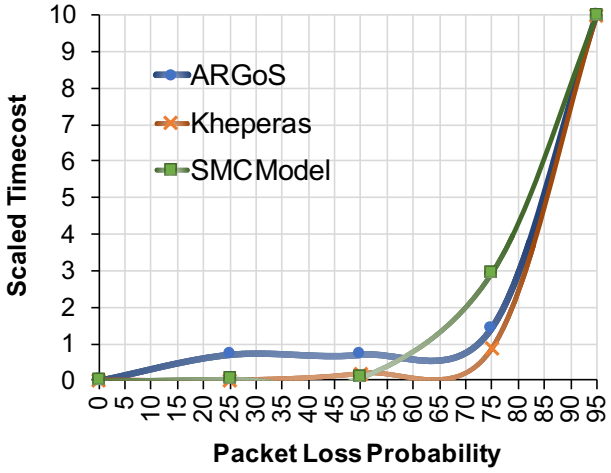


Figure 4.14 Scaled timecost for an elect leader scenario: Comparison between the SMC Model, ARGoS and real-world experiment with 5 Khepera group

Figures 4.16(a) and 4.16(b) show swarm-based model performance in terms of verification time and memory usage respectively. These figures show that the model can check a swarm composed of up to 1000 agents fully connected (connectivity rate 100 %) in few minutes with a memory usage rate close to 1 GB. Moreover, our model can verify 2000-agent swarms in about 1h using at most 4GB of memory for the most pessimistic case (*i.e.* 100% of connectivity). Here, we note a clear decrease, with respect to the first model in the memory consumption as well as in the computing time needed for verification. Contrary to the agent-based model, this swarm-based model can handle swarms composed of up to 5000 agents fully connected with reasonable amounts of memory usage and computing time.

It is important to note that the computing time can be also reduced by using Distributed SMC UPPAAL [4]. This tool allows a faster verification time by using several computers working in parallel under a master/slaves architecture.

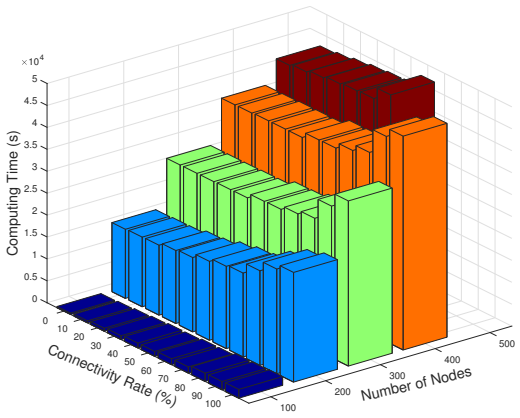
4.7 Conclusions

The verification and the validation of autonomous systems have become a very active area of research. In this paper, we verify and assess the robustness of consensus-based behaviors in a networked robotic swarm. We use formal methods which are known by their effectiveness and accuracy in checking real time systems. More precisely, we propose a Stochastic Model Checking approach to verify the consensus based strategies in terms of the communication quality and robot failure probability. Our formal model is tested and validated on two common scenarios (leader election and auction-based task allocation), with satisfactory results. The assessment of the communication quality impact was comparable to what was recorded using physics-based simulations and real-world experiments. To deal with state explosion problem, two abstractions are proposed and evaluated in term of space and time complexity, according to the network size and connectivity rate: we show that the model can deal with large number of robots in a reasonable time on a standard laptop.

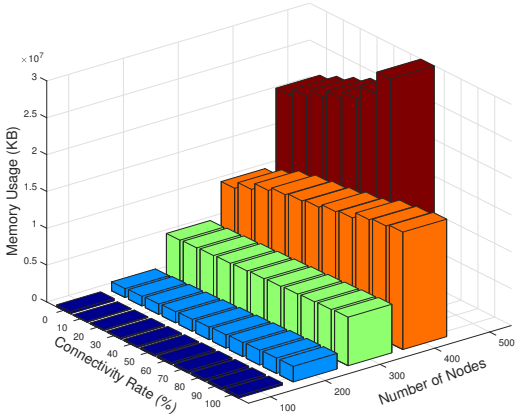
Our formal framework is based on a three-step strategy to model and assess consensus-based behaviors of collective robotic systems. First, the designer uses the automata system to formally model the swarm. Second, they implement the update function to specify the consensus primitives of interest. And finally, the designer specifies the system requirements and uses SMC to analyze the model and uncover any flaws.

As future work, the approach can be applied to more complex case studies where conflicts may occur during the consensus process.

[4] http://people.cs.aau.dk/ adavid/smc/cases.html

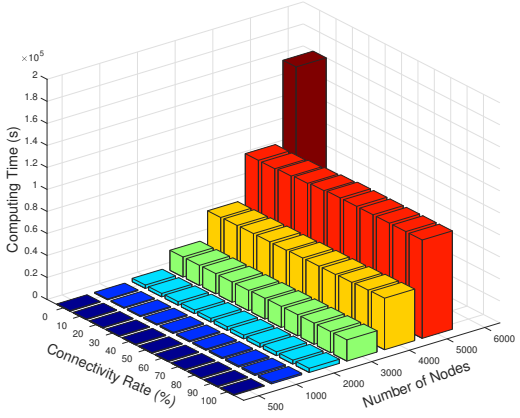


(a) Computing time for the agent-based model

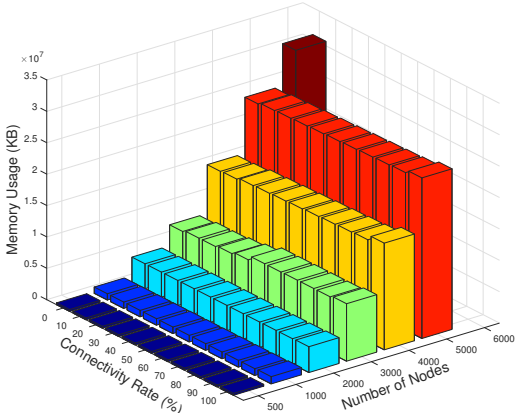


(b) Memory usage for the agent-based model

Figure 4.15 Performance of the agent-based model.



(a) Computing time for the swarm-based model



(b) Memory usage for the the swarm-based model

Figure 4.16 Performance of the swarm-based model.

CHAPTER 5 GENERAL DISCUSSION

This chapter discusses the research work presented in the Chapters 3 and 4 and underlines its impact and potential to alter the landscape offered by the exiting literature. The discussion is divided into three sections to cover the research areas of interest —path planning for mobile robots, artificial intelligence and swarm robotics—.

5.1 On Path Planning and Robots Navigational Capabilities

Path planning is an important primitive for autonomous mobile robots that are intended to navigate their environments in an optimal way, whether that be with a minimal traveling distance, time, turning, braking or anything a specific application requires. A special interest has been granted to solutions that meet the specific real-time constraints –referred to as responsiveness– of many robotic applications that require a robot to navigate an environment while avoiding obstacles. Research has produced many navigation algorithms and a lot of effort has been put forth to specifically afford an acceptable level of responsiveness. We elaborated on this need by providing a real-time navigational algorithm. Our main findings came from the work presented in Chapter 3 and submitted to Science.

In Chapter 1, we enumerate our research objectives which include the desire to conceive a real-time navigation system (Objective 1). The design takes into consideration the path planning problem and promotes a solution that associates two key features, namely path optimality (*i.e.* shortest path) and responsiveness. The research in literature, prior to this work, typically focused on algorithms that tackle one of these features at a time. For example, [16] and [17] are reviews for some path planners (GBP and SBP) that account more for path optimality or responsiveness respectively.

Being inspired by APF-based planners [18], our solution (see Chapter 3) makes use of a virtual magnetic field to provide an efficient navigation system. The virtual magnetic field is modeled by well-established physics-based equations, i.e. Maxwell's equations. It has two key properties that make it the best candidate to solve a path planning problem: i) it occupies all the environment but the obstacles, and ii) it optimally matches any point in the environment to the source (goal position) whenever a feasible way exists, making it possible to mitigate the inherent problem of APF *i.e.* local minima. Compared to other approaches including APF algorithms, the solution, that we propose, is able to guarantee an optimal path whenever one exists. The algorithm is very responsive and is able to run at high speed level (up to 200 frames per second). Consequently, it: i) can adapt to both uncertain and incomplete information in dynamic environments, ii) has an intuitive ability to

re-plan paths, iii) is suitable for multi-query tasks where many goals are defined simultaneously, and iv) does not need any post processing stage unlike RRT-based approaches which require an iterative processing to smooth the path.

Magnetic potential fields were used in [89] to perform path planning. The authors have implemented the differential equations using Finite Difference Method FDM [88]. Although, they succeeded to show the ability of the natural magnetic field to avoid APF problems, the approach cannot outperform the state of the art because of its heavy implementation. Elaborating on that, we devised a learning-based strategy to solve these equations in real-time, named MaxConvNet (details in Chapter 3). The magnetic potential field modeled by Maxwell's equations completely eliminates the local minima problem, which is exhibited in most APF methods. It is easily extendable to 3D; the time dimension is modeled by default which means it is a suitable model for time-varying environments.

The analysis in Section 3.2 shows how MaxConvNet can be applied to perform autonomous navigation. We supported our findings by Videos S1 to S5. The videos are in `https://drive.google.com/open?id=1vU5ruAQCW8wbH4nMESxZWosBvu6SCO5j` and their captions can be found in Section 3.5.3. First, we conducted a set of physics-based simulations whereby we implement MaxConvNet on different vehicles (*i.e.* drone and Sport Utility Vehicle (SUV)) to control their motion in different environments (see Videos S1-2). Furthermore, we used MaxConvNet to control a Traxxas Stampede rover equipped with Ydlidar X4 lidar and a PixRacer. We run a set of experiments outdoors as shown Video S4. Moreover, we illustrated how MaxConvNet can be applied to videogame technology: we implemented a game-based scenario where an AI tries to catch a human-controlled player using MaxConvNet in a highly dynamic environment. The position of the player is encoded as the goal and used along with the obstacle information by the AI to predict a feasible path to track the player (see Video S3 for an example).

In section 3.2.3, we conducted an experimental study to compare MaxConvNet and RRT*-based approach in terms of path cost and run time. The findings pointed out that MaxConvNet approach is more efficient that the SBP since it: i) is much faster, ii) does not rely on randomness; therefore, a single optimal path is provided for a given scenario, iii) has a stable runtime as it does not depend on neither the scene complexity nor the computed path length.

Nevertheles, SBP and particularly RRT*-based approaches can scale to n-dimensional Search Space. Our MaxConvNet model as it is entended to solve Unmanned Vehicles (UVs) planning problems can handle up to 3-dimensional spaces

5.2 On Artificially Intelligent Autonomous Vehicles

Not too long ago, all industrial robots could only be programmed to carry out a defined sequence of instructions that do not require any interactive capabilities of the robot with its environment. The robots are, therefore, quite limited in their functionality and require a sustained human support. The ability of the robot to interact with its surroundings and autonomously decide its future maneuvers based on its acquired knowledge is of utmost importance. It allows the robot to perform more complex tasks and limits the need for human intervention. A captivating interest has exploded in the past decade about the future of Artificial Intelligence (AI) and its impact of the robotic field. Most researchers agree on AI's ability to support a new generation of intelligent robots that are able to interact with the environment, learn, reason and plan. The results presented in this area originated from the research detailed in Chapter 3.

The problem definition in Chapter 1 identified the lack "a way to engineer bio-inspired systems to produce intelligent navigation solutions". The research effort produced in this field, and presented in Chapter 3, aims at bridging the gap between the theory inspired from nature and robotics by employing AI. That is, we promote for an AI-based program that employs natural intelligence techniques and physical laws to fulfill the need of an efficient navigation system —in particular with respect to the real-time constraints.

The core idea of the proposed navigation system is inspired from the animals' kingdom where some species are able to sense the terrestrial magnetic field and use it as natural compass to navigate their environment. Starting from this idea, we modeled path planning as a physics-based problem. As stated in the previous section, the operating environment of a mobile robots is modeled as a distribution of virtual magnetic waves using Maxwell's equations. This magnetic distribution is used to determine an optimal path using a gradient-based optimization algorithm.

To endow mobile robots with the ability of predicting the virtual magnetic field distribution of their environment, we propose an AI-based real-time solver of Maxwell's equations that can be applied not only for robotic navigation but also for embedded system applications with limited computing capabilities such as mobile games. Our AI solution works by automatically recognizing the distribution of the magnetic field in a 2D time varying environment defined as a conductivity map, whereby electrical currents are supposed to be floating. The currents are attracted by a goal position which is assigned to a very high conductivity and they are repulsed by the obstacles which are assigned to zero conductivity. The environment is assigned to a degraded conductivity *i.e.* the closer to the goal the higher the conductivity. The goal with the highest conductivity acts as a sink to the currents flow and induces an intense magnetic field, that spreads everywhere in the whole environment, whereas the obstacles behave as insulators.

The AI solver is implemented as a deep learning model using auto-encoder convolutional neural networks that predicts the magnetic field distribution in any 2D cluttered environment. The model is trained *exclusively* in simulation and does not need transfer learning because it is already based on an abstraction of the environment.

In Section 3.2.2, we discussed the ability of the model to make correct predictions for unknown environments. Results report that the model error on the testing dataset is less than -25dB. This means that it is able to produce a magnetic field distribution with 95 % of accuracy at the pixel level. Moreover, results show that the model is able to tackle perfectly the problem of local minima and it outperforms the state of the art navigation algorithms by ensuring completeness and providing an optimal path in real-time fashion (*i.e.* up to 200 fps). The low computational cost of MaxConvNet (see Section 3.2.4) allows implementations on small, low-cost, single-board computers as the Jetson Nano or the Raspberry Pi, paving the way for pervasive robot applications.

The proposed method can be readily extended to other navigation problems, such as flocking. Video S5 show a scenario where MaxConvNet is used to control the motion of 5 drones simultaneously.

5.3 On Multi-agent Systems and Swarm Robotics

A key feature of autonomous multi-agent and swarm robotic systems is their ability to coordinate collective behaviors by handling the flow of exchanged information and maintaining connectivity. That is the robustness of the communication process is a property of great interest for the study of the challenges related to those complex structures – including sensors interference and situations where communication bandwidths are severely limited. The results, in this area of research, are collected into an article presented in Chapter 4. The main contribution of the article is the implementation of a formal framework that assesses the communication robustness of a swarm robotic system. We aim at addressing the lack of "a formal definition of what a robust communication is and what robustness entails" and "a formal model and methodology to assess and quantify robustness", as identified in Chapter 1.

An important part of our work is the modeling of the *timing properties* of a set of consensus-based primitives while considering connectivity issues and fallibility of robots. The value of our work consists of the ability of the model to validate the correctness and the robustness of consensus-based behaviors in collective robotic systems. This has been done by using i) probabilistic transitions to simulate the effect of some networking metrics such as packet loss and robot failures, and ii) quantitative measurable properties to accurately express the system requirements.

Two SMC-based models have been proposed with different abstraction levels. The agent-based model allows a better understanding of the consensus protocol of interest. However, it is less scal-

able due to the state explosion problem. The swarm-based model abstracts the agents' interactions and focuses on the consensus status evolution. This model is less expressive but exhibits better scalability. Although the models exhibit two different abstraction levels, they have the same dynamics and are able to produce the same statistical results.

Results in Section 4.6 show that the agent-based model is able to handle relatively large swarms (> 300 agents) but it still suffers from the combinatorial explosion problem for large and fully connected networks. The second model greatly reduces the explosion problem. It is able to deal with up to 2000 fully-connected robots using as little as 4 GB of memory and returns results in a few minutes. Moreover, the revised model can analyze larger swarms (up to 5000 fully-connected robots) using at most 24GB of memory. However, the model exhibits longer computing time, possibly several hours. This can be greatly reduced using the distributed version of SMC UPPAAL, which makes use of OpenMPI to parallelize the verification process over several machines using a master/slave architecture.

In this work, the dynamic aspect (*i.e.* robot mobility) has been abstracted as the properties of interest are not concerned with the exact position of each robot. That is, the focus is shifted to the global (abstract) behavior of the overall system. This allows us to provide a scalable model that accounts for a huge number of robots (2000+). Nevertheless, it is worth noting, that we are able to simulate some system dynamics by considering the fact that robots may fail which is equivalent to robots going out of communication range.

CHAPTER 6 CONCLUSION

The research work, undertaken in this dissertation, started with the aim to i) study present challenges that each wheeled mobile robot needs to overcome to become autonomous, ii) implement methodologies that tackle those challenges, and iii) set up an efficent navigation system for mobile robots that can scale up to handle not only solo and also cooperative navigation.

The two main research objectives identified in Chapter 1 were accomplished through the two research articles included in the body of this dissertation (Chapters 3 and 4). In particular, ***objective 1*** – propose a bio-inspired navigation system that acounts for responsiveness and scalability–, is undertaken in chapter 3, and ***objective 2*** – implement a methodology to enhance the eventual implementation of *objective 1* in the context of swarm robotic system–, has been addressed in Chapter 4. The previous chapter discussed our findings and their impact in the fields of robotic navigation, artificial intelligence and swarm robotic systems. In this concluding chapter, we wrap up the thesis and present directions for the prospective future work.

6.1 Final Remarks

One of the most significant problems of mobile-robot systems whether they are single-robot-based or multi-robot-based, revolves around autonomous navigation which requires that a vehicle is able to plan its path and execute its plan without human intervention.

A navigation system that acounts for robot-environment interaction and learning would provide a robot with a genuine ability to reason under environmental uncertainty and act accordingly. This feature is actually at utmost importance to ensure that the robot can behave autonomously and sustainably. In the light of this, a huge research work has been put forth to address autonomous navigation taking into consideration the other strongly-related modules such as perception, cognition, localization and human-robot interaction. At the same time, a growing interest has been observed in cooperative approaches, whereby communication robustness is considered as a vital concern.

In this context, the scientific contributions behind this thesis can be summarized as follows :

1. The design of a physics-based mapping tool that allows to model any 2D time-varying environment as distribution of a virtual magnetic field using well-established Maxwell's equations;

2. The design and implementation of a generic deep learning framework able to solve Maxwell's

equations. That is to say, it is able to predict a correct magnetic field distribution given an environmental map generated using the aforementioned physics-based mapping tool;

3. The implementation of a gradient-based path planner that allows to determine an optimal path from the previously computed magnetic distribution;

4. Insights on the robustness of collective behaviors by providing a formal framework to assess and verify the correctness of consensus-based behaviors for swarm-robotic systems;

5. A proof of concept of how the proposed path planner can be used to perform a collective behavior such as flocking.

6.2 Future Research and Improvements

The results presented throughout this dissertation point out to several interesting directions for future work.

In the context of robot navigation. It would be interesting to:

— Implement a data fusion system able to couple different sensors data such as Global Positioning System (GPS), Inertial Navigation System (INS), lidar and camera to improve perception and consequently the planning results;

— Adapt the navigation solution to indoor applications by integrating techniques such as Simultaneous Localization and Mapping (SLAM) that allows a mobile robot to automatically correct its position and orientation;

— Extend the path planning technique to 3D to enable a more efficient navigation for Unmanned Aerial Vehicles (UAVs). Then, the deep neural framework should be trained with a large amount of environments data including different shapes of obstacles;

— Implement the navigation technique on multi-robot systems. Vision data shoud be coupled with the shared data at the robot level. This would be necessary to improve and enhance the accuracy of the technique and achieve an efficient cooperative navigation;

— Take into account of the dynamics of the system in the conductivity distribution;

— Extend the method to perform pattern formation tasks using virtual magnetic field.

In the context of ensuring the correctness of swarm robotic behaviors. It would be interesting to:

— Integrate the ability of the swarm to move and interact with its environment to generalize the impact of communication and connectivity issues on more complex and dynamic emergent behaviors;

— Generalize the approach to more complex case studies where conflicts may occur during the consensus process.

REFERENCES

[1] Y. Dobrev, M. Vossiek, M. Christmann, I. Bilous, and P. Gulden, "Steady delivery: Wireless local positioning systems for tracking and autonomous navigation of transport vehicles and mobile robots," *IEEE Microwave Magazine*, vol. 18, no. 6, pp. 26–37, 2017.

[2] A. Ollero, A. Viguria, M. A. Trujillo, M. Oetiker, and B. Revaz, "Inspection and maintenance," *Aerial Robotic Manipulation*, pp. 367–379, 2019.

[3] S. Witwicki, J. C. Castillo, J. Messias, J. Capitan, F. S. Melo, P. U. Lima, and M. Veloso, "Autonomous surveillance robots: A decision-making framework for networked muiltiagent systems," *IEEE Robotics & Automation Magazine*, vol. 24, no. 3, pp. 52–64, 2017.

[4] A. Jeremic and A. Nehorai, "Design of chemical sensor arrays for monitoring disposal sites on the ocean floor," *IEEE Journal of Oceanic Engineering*, vol. 23, no. 4, pp. 334–343, 1998.

[5] D. M. Gordon, "The ecology of collective behavior," *Plos biology*, vol. 12, no. 3, p. e1001805, 2014.

[6] M. Yomosa, T. Mizuguchi, G. Vásárhelyi, and M. Nagy, "Coordinated behaviour in pigeon flocks," *Plos one*, vol. 10, no. 10, p. e0140558, 2015.

[7] S. Garnier, M. Combe, C. Jost, and G. Theraulaz, "Do ants need to estimate the geometrical properties of trail bifurcations to find an efficient route? a swarm robotics test bed," *PLoS computational biology*, vol. 9, no. 3, p. e1002903, 2013.

[8] L.-A. Poissonnier, S. Motsch, J. Gautrais, J. Buhl, and A. Dussutour, "Experimental investigation of ant traffic under crowded conditions," *eLife*, vol. 8, 2019.

[9] L.-Q. Wu and J. D. Dickman, "Neural correlates of a magnetic sense," *Science*, vol. 336, no. 6084, pp. 1054–1057, 2012.

[10] H. McCreery, "A comparative approach to cooperative transport in ants: individual persistence correlates with group coordination," *Insectes sociaux*, vol. 64, no. 4, pp. 535–547, 2017.

[11] S. Knorn, Z. Chen, and R. H. Middleton, "Overview: Collective control of multiagent systems," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 4, pp. 334–347, 2016.

[12] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015.

[13] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," in *International workshop on swarm robotics*. Springer, 2004, pp. 10–20.

[14] M. N. Rastgoo, B. Nakisa, M. F. Nasrudin, A. Nazri, and M. Zakree, "A critical evaluation of litterature on robot path planning in dynamic environment," *Journal of Theoretical & Applied Information Technology*, vol. 70, no. 1, 2014.

[15] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, pp. 171–185, 2018.

[16] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles." *IEEE Trans. Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[17] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[18] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Autonomous robot vehicles*, pp. 396–404, 1986.

[19] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.

[20] R. Jarvis, "Distance transform based path planning for robot navigation," in *Recent Trends in Mobile Robots*. World Scientific, 1993, pp. 3–31.

[21] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Department of Computer Science; Iowa State University, Tech. Rep., 1998.

[22] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[23] F. Islam, J. Nasir, U. Malik, Y. Ayaz, and O. Hasan, "RRT*-smart: Rapid convergence implementation of RRT* towards optimal solution," *Mechatronics and Automation (ICMA), 2012 International Conference on*, pp. 1651–1656, 2012.

[24] M. Otte and E. Frazzoli, "Rrt^X: Real-time motion planning replanning for environments with unpredictable obstacles," *Algorithmic Foundations of Robotics XI*, pp. 461–478, 2015.

[25] J. Bruce and M. M. Veloso, "Real-time randomized path planning for robot navigation," *Robot Soccer World Cup*, pp. 288–295, 2002.

[26] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[27] K. Naderi, J. Rajamäki, and P. Hämäläinen, "Rt-rrt*: a real-time path planning algorithm based on rrt," *Proc. of the 8th ACM SIGGRAPH Conference on Motion in Games*, pp. 113–118, 2015.

[28] P. Szulczyński, D. Pazderski, and K. Kozłowski, "Real-time obstacle avoidance using harmonic potential functions," *Journal of Automation Mobile Robotics and Intelligent Systems*, vol. 5, pp. 59–66, 2011.

[29] Y. Liu and Y. Zhao, "A virtual-waypoint based artificial potential field method for uav path planning," *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, pp. 949–953, 2016.

[30] Y.-b. Chen, G.-c. Luo, Y.-s. Mei, J.-q. Yu, and X.-l. Su, "Uav path planning using artificial potential field method updated by optimal control theory," *International Journal of Systems Science*, vol. 47, no. 6, pp. 1407–1420, 2016.

[31] F. Bayat, S. Najafinia, and M. Aliyari, "Mobile robots path planning: Electrostatic potential field approach," *Expert Systems with Applications*, vol. 100, pp. 68–78, 2018.

[32] R. Raja, A. Dutta, and K. Venkatesh, "New potential field method for rough terrain path planning using genetic algorithm for a 6-wheel rover," *Robotics and Autonomous Systems*, vol. 72, pp. 295–306, 2015.

[33] O. Montiel, U. Orozco-Rosas, and R. Sepúlveda, "Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles," *Expert Systems with Applications*, vol. 42, no. 12, pp. 5177–5191, 2015.

[34] J. P. C. Tuazon, K. G. V. Prado, N. J. A. Cabial, R. L. Enriquez, F. L. C. Rivera, and K. K. D. Serrano, "An improved collision avoidance scheme using artificial potential field with fuzzy logic," *2016 IEEE Region 10 Conference (TENCON)*, pp. 291–296, 2016.

[35] T. Y. Abdalla, A. A. Abed, and A. A. Ahmed, "Mobile robot navigation using pso-optimized fuzzy artificial potential field with fuzzy control," *Journal of Intelligent & Fuzzy Systems*, vol. 32, no. 6, pp. 3893–3908, 2017.

[36] O. Wahyunggoro, A. I. Cahyadi *et al.*, "Quadrotor path planning based on modified fuzzy cell decomposition algorithm." *Telkomnika*, vol. 14, no. 2, 2016.

[37] M. J. Bency, A. H. Qureshi, and M. C. Yip, "Neural path planning: Fixed time, near-optimal path generation via oracle imitation," *arXiv preprint arXiv:1904.11102*, 2019.

[38] T. Dang, S. Khattak, C. Papachristos, and K. Alexis, "Anomaly detection and cognizant path planning for surveillance operations using aerial robots," *International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 667–673, 2019.

[39] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, "End-to-end interpretable neural motion planner," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8660–8669.

[40] Y. Tan, "A survey on swarm robotics," in *Handbook of Research on Design, Control, and Modeling of Swarm Robotics*. IGI Global, 2016, pp. 1–41.

[41] A. V. Savkin, C. Wang, A. Baranzadeh, Z. Xi, and H. T. Nguyen, "Distributed formation building algorithms for groups of wheeled mobile robots," *Robotics and Autonomous Systems*, vol. 75, pp. 463–474, 2016.

[42] A. E. Turgut, H. Çelikkanat, F. Gökçe, and E. Şahin, "Self-organized flocking in mobile robot swarms," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 97–120, 2008.

[43] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, Mar 2013.

[44] A. J. Sharkey and N. Sharkey, "The application of swarm intelligence to collective robots," in *Advances in applied artificial intelligence*. IGI Global, 2006, pp. 157–185.

[45] V. Costa, M. Duarte, T. Rodrigues, S. M. Oliveira, and A. L. Christensen, "Design and development of an inexpensive aquatic swarm robotics system," in *OCEANS 2016-Shanghai*. IEEE, 2016, pp. 1–7.

[46] E. Trulls, A. Corominas Murtra, J. Pérez-Ibarz, G. Ferrer, D. Vasquez, J. M. Mirats-Tur, and A. Sanfeliu, "Autonomous navigation for mobile service robots in urban pedestrian environments," *Journal of Field Robotics*, vol. 28, no. 3, pp. 329–354, 2011.

[47] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[48] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): Part ii," *IEEE robotics & automation magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[49] K. J. O'Hara, D. B. Walker, and T. R. Balch, "Physical path planning using a pervasive embedded network," *IEEE Transactions on Robotics*, vol. 24, no. 3, pp. 741–746, 2008.

[50] G. Beni, "From swarm intelligence to swarm robotics," in *International Workshop on Swarm Robotics*. Springer, 2004, pp. 1–9.

[51] N. Palmieri, X.-S. Yang, F. De Rango, and S. Marano, "Comparison of bio-inspired algorithms applied to the coordination of mobile robots considering the energy consumption," *Neural Computing and Applications*, vol. 31, no. 1, pp. 263–286, 2019.

[52] G. Lee and N. Y. Chong, "Flocking controls for swarms of mobile robots inspired by fish schools," in *Recent advances in multi robot systems*. IntechOpen, 2008.

[53] N. Palmieri and S. Marano, "Discrete firefly algorithm for recruiting task in a swarm of robots," in *Nature-Inspired Computation in Engineering*. Springer, 2016, pp. 133–150.

[54] S. Bernardi, A. Colombi, and M. Scianna, "A particle model analysing the behavioural rules underlying the collective flight of a bee swarm towards the new nest," *Journal of biological dynamics*, vol. 12, no. 1, pp. 632–662, 2018.

[55] L. A. Taylor, G. K. Taylor, B. Lambert, J. A. Walker, D. Biro, and S. J. Portugal, "Birds invest wingbeats to keep a steady head and reap the ultimate benefits of flying together," *PLoS biology*, vol. 17, no. 6, p. e3000299, 2019.

[56] J. E. Boström, S. Åkesson, and T. Alerstam, "Where on earth can animals use a geomagnetic bi-coordinate map for navigation?" *Ecography*, vol. 35, no. 11, pp. 1039–1047, 2012.

[57] E. C. Ferrer, "The blockchain: a new framework for robotic swarm systems," in *Proceedings of the Future Technologies Conference*. Springer, 2018, pp. 1037–1058.

[58] H. Yuasa and M. Ito, "Coordination of many oscillators and generation of locomotory patterns," *Biological Cybernetics*, vol. 63, no. 3, pp. 177–184, 1990.

[59] M. Masár and I. Budinská, "Robot coordination based on biologically inspired methods," in *Advanced Materials Research*, vol. 664. Trans Tech Publ, 2013, pp. 891–896.

[60] M. Dorigo, M. Birattari, and M. Brambilla, "Swarm robotics," *Scholarpedia*, vol. 9, no. 1, p. 1463, 2014.

[61] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, E. Bonabeau, and G. Theraula, *Self-organization in biological systems*. Princeton university press, 2003, vol. 7.

[62] M. Senanayake, I. Senthooran, J. C. Barca, H. Chung, J. Kamruzzaman, and M. Murshed, "Search and tracking algorithms for swarms of robots: A survey," *Robotics and Autonomous Systems*, vol. 75, pp. 422–434, 2016.

[63] V. Strobel, E. Castelló Ferrer, and M. Dorigo, "Managing byzantine robots via blockchain technology in a swarm robotics collective decision making scenario," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 541–549.

[64] L. Brunet, H.-L. Choi, and J. P. How, "Consensus-based auction approaches for decentralized task assignment," in *AIAA Guidance, Navigation, and Control Conference, Honolulu, Hawaii*, 2008, p. 6839.

[65] Y. Kuriki and T. Namerikawa, "Experimental validation of cooperative formation control with collision avoidance for a multi-uav system," in *Automation, Robotics and Applications (ICARA), 2015 6th International Conference on*. IEEE, 2015, pp. 531–536.

[66] C. Pinciroli, A. Lee-Brown, and G. Beltrame, "A tuple space for data sharing in robot swarms," in *BICT 2015, Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), New York City, United States, December 3-5, 2015*, 2015, pp. 287–294.

[67] X. Wei, D. Fengyang, Z. Qingjie, Z. Bing, and S. Hongchang, "A new fast consensus algorithm applied in rendezvous of multi-uav," in *Control and Decision Conference (CCDC), 2015 27th Chinese*. IEEE, 2015, pp. 55–60.

[68] A. Shia, F. B. Bastani, and I.-L. Yen, "A highly resilient framework for autonomous robotic swarm systems operating in unknown, hostile environments," in *2011 Tenth International Symposium on Autonomous Decentralized Systems*. IEEE, 2011, pp. 147–153.

[69] M. Farrell, M. Luckcuck, and M. Fisher, "Robotics and integrated formal methods: necessity meets opportunity," *International Conference on Integrated Formal Methods*, pp. 161–171, 2018.

[70] S. Konur, C. Dixon, and M. Fisher, "Analysing robot swarm behaviour via probabilistic model checking," *Robotics and Autonomous Systems*, vol. 60, no. 2, pp. 199–213, 2012.

[71] L. Mikaël, "Formal verification of flexibility in swarm robotics," Ph.D. dissertation, Citeseer, 2012.

[72] C. Dixon, A. F. Winfield, M. Fisher, and C. Zeng, "Towards temporal verification of swarm robotic systems," *Robotics and Autonomous Systems*, vol. 60, no. 11, pp. 1429–1441, 2012.

[73] A. F. Winfield, J. Sa, M.-C. Fernández-Gago, C. Dixon, and M. Fisher, "On formal specification of emergent behaviours in swarm robotic systems," *International journal of advanced robotic systems*, vol. 2, no. 4, p. 39, 2005.

[74] G. Behrmann, A. David, K. G. Larsen, J. Hakansson, P. Petterson, W. Yi, and M. Hendriks, "Uppaal 4.0," in *Quantitative Evaluation of Systems, 2006. QEST 2006. Third International Conference on*. IEEE, 2006, pp. 125–126.

[75] P. Kouvaros and A. Lomuscio, "Verifying fault-tolerance in parameterised multi-agent systems." in *IJCAI*, 2017, pp. 288–294.

[76] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey," *Journal of intelligent and robotic systems*, vol. 53, no. 3, pp. 263–296, 2008.

[77] D. Sorin and G. Konidaris, "Enabling faster, more capable robots with real-time motion planning," 2018, https://spectrum.ieee.org/automaton/robotics/robotics-software/enabling-faster-more-capable-robots-with-real-time-motion-planning, last accessed on 2019-10-14.

[78] C. Fisher, "Waymo's fully-automated shuttles are picking up riders around phoenix," 2019, https://www.engadget.com/2019/10/28/waymo-rider-only-autonomous-taxis-phoenix/, last accessed on 2019-10-24.

[79] F. Jon, "Skydio's station lets self-flying drones work around the clock," 2019, https://www.engadget.com/2019/10/16/skydio-2-dock-autonomous-drone/, last accessed on 2019-10-24.

[80] D. Barrie, *Supernavigators: Exploring the wonders of how animals find their way*. The Experiment, 2019.

[81] J. Davis, "Mathematical modeling of earth's magnetic field," *Technical Note, Virginia Tech, Blacksburg*, 2004.

[82] S. Edwards, C. Parnell, L. Harra, J. Culhane, and D. Brooks, "A comparison of global magnetic field skeletons and active-region upflows," *Solar Physics*, vol. 291, no. 1, pp. 117–142, 2016.

[83] W. H. Campbell, *Earth magnetism: a guided tour through magnetic fields*. Elsevier, 2001.

[84] L. Chang, "Progress, contribution and challenges of earth-magnetism navigation," *Automation, Control and Intelligent Systems*, vol. 5, no. 1, p. 8, 2017.

[85] G. Huang, B. K. Taylor, and D. Akopian, "A low-cost approach of magnetic field-based location validation for global navigation satellite systems," *IEEE Transactions on Instrumentation and Measurement*, 2019.

[86] J. R. Cary and S. G. Shasharina, "Omnigenity and quasihelicity in helical plasma confinement systems," *Physics of Plasmas*, vol. 4, no. 9, pp. 3323–3333, 1997.

[87] ——, "Helical plasma confinement devices with good confinement properties," *Physical review letters*, vol. 78, no. 4, p. 674, 1997.

[88] O. C. Zienkiewicz, R. L. Taylor, P. Nithiarasu, and J. Zhu, *The finite element method*. McGraw-hill London, 1977, vol. 36.

[89] A. M. Hussein and A. Elnagar, "Motion planning using maxwell's equations," *Intelligent Robots and Systems*, vol. 3, pp. 2347–2352, 2002.

[90] J. G. Van Bladel, *Electromagnetic fields*. John Wiley & Sons, 2007, vol. 19.

[91] P. G. Huray, *Maxwell's equations*. John Wiley & Sons, 2011.

[92] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27–48, 2016.

[93] X. Guo, W. Li, and F. Iorio, "Convolutional neural networks for steady flow approximation," *International Conference on Knowledge Discovery and Data Mining*, pp. 481–490, 2016.

[94] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[95] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, 2015.

[96] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[97] M. I. Meyer, A. Galdran, A. M. Mendonça, and A. Campilho, "A pixel-wise distance regression approach for joint retinal optical disc and fovea detection," *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 39–47, 2018.

[98] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[99] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proc. International Conference on Learning Representations (ICLR)*, 2015.

[100] M. N. Ã-zisik, M. N. Özısık, and M. N. Özışık, *Heat conduction*. John Wiley & Sons, 1993.

[101] N. Endou, K. Narita, and Y. Shidama, "The lebesgue monotone convergence theorem," *Formalized Mathematics*, vol. 16, no. 2, pp. 167–175, 2008.

[102] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," *Field and service robotics*, pp. 621–635, 2018.

[103] X. Zhang, Y. Wang, and W. Shi, "pcamp: Performance comparison of machine learning packages on the edges," *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, Jul. 2018. [Online]. Available: https://www.usenix.org/conference/hotedge18/presentation/zhang

[104] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," *Proc. of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

[105] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *Proc. International Conference on Machine Learning (ICML)*, pp. 448–456, 2015.

[106] D. Han, "Comparison of commonly used image interpolation methods," *Proc. of the 2nd International Conference on Computer Science and Electronics Engineering*, 2013.

[107] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[108] B. Bilgic, I. Chatnuntawech, A. P. Fan, K. Setsompop, S. F. Cauley, L. L. Wald, and E. Adalsteinsson, "Fast image reconstruction with l2-regularization," *Journal of magnetic resonance imaging*, vol. 40, no. 1, pp. 181–191, 2014.

[109] A. Van Ooyen and B. Nienhuis, "Improving the convergence of the back-propagation algorithm," *Neural networks*, vol. 5, no. 3, pp. 465–471, 1992.

[110] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *Proc. of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[111] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning," *OSDI*, vol. 16, pp. 265–283, 2016.

[112] C. W. Reynolds, "Steering behaviors for autonomous characters," *Game developers conference*, vol. 1999, pp. 763–782, 1999.

[113] F. Rossi, S. Bandyopadhyay, M. Wolf, and M. Pavone, "Review of multi-agent algorithms for collective behavior: a structural taxonomy," *arXiv preprint arXiv:1803.05464*, 2018.

[114] C. Robin and S. Lacroix, "Multi-robot target detection and tracking: taxonomy and survey," *Autonomous Robots*, vol. 40, no. 4, pp. 729–760, Apr 2016.

[115] R. De Nicola and E. Kühn, *Software Engineering and Formal Methods*. Springer, 2016.

[116] G. P. Das, T. M. McGinnity, S. A. Coleman, and L. Behera, "A distributed task allocation algorithm for a multi-robot system in healthcare facilities," *Journal of Intelligent & Robotic Systems*, vol. 80, no. 1, pp. 33–58, 2015.

[117] P. Kouvaros and A. Lomuscio, "Formal verification of opinion formation in swarms," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 1200–1208.

[118] E. M. Clarke, O. Grumberg, and D. Peled, *Model checking*. MIT press, 1999.

[119] M. Kwiatkowska, G. Norman, and D. Parker, "Prism 4.0: Verification of probabilistic real-time systems," in *Computer aided verification*. Springer, 2011, pp. 585–591.

[120] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "Argos:

a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[121] H. Barringer, M. Fisher, D. M. Gabbay, and G. Gough, *Advances in temporal logic*. Springer Science & Business Media, 2013, vol. 16.

[122] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "Nusmv 2: An opensource tool for symbolic model checking," in *International Conference on Computer Aided Verification*. Springer, 2002, pp. 359–364.

[123] S. Amin, A. Elahi, K. Saghar, and F. Mehmood, "Formal modelling and verification approach for improving probabilistic behaviour of robot swarms," in *Applied Sciences and Technology (IBCAST), 2017 14th International Bhurban Conference on*. IEEE, 2017, pp. 392–400.

[124] E. Gjondrekaj, M. Loreti, R. Pugliese, F. Tiezzi, C. Pinciroli, M. Brambilla, M. Birattari, and M. Dorigo, "Towards a formal verification methodology for collective robotic systems," in *International Conference on Formal Engineering Methods*. Springer, 2012, pp. 54–70.

[125] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Progress on the state explosion problem in model checking," in *Informatics*. Springer, 2001, pp. 176–194.

[126] B. Yang, R. E. Bryant, D. R. O'Hallaron, A. Biere, O. Coudert, G. Janssen, R. K. Ranjan, and F. Somenzi, "A performance study of bdd-based model checking," in *International Conference on Formal Methods in Computer-Aided Design*. Springer, 1998, pp. 255–289.

[127] A. Armando, J. Mantovani, and L. Platania, "Bounded model checking of software using smt solvers instead of sat solvers," *International Journal on Software Tools for Technology Transfer*, vol. 11, no. 1, pp. 69–83, 2009.

[128] E. M. Clarke, O. Grumberg, and D. E. Long, "Model checking and abstraction," *ACM transactions on Programming Languages and Systems (TOPLAS)*, vol. 16, no. 5, pp. 1512–1542, 1994.

[129] G. Holzmann, *Spin model checker, the: primer and reference manual*. Addison-Wesley Professional, 2003.

[130] A. Hessel, K. G. Larsen, M. Mikucionis, B. Nielsen, P. Pettersson, and A. Skou, "Testing real-time systems using uppaal," in *Formal methods and testing*. Springer, 2008, pp. 77–117.

[131] G. Agha and K. Palmskog, "A survey of statistical model checking," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 28, no. 1, p. 6, 2018.

[132] A. Tartakovsky, I. Nikiforov, and M. Basseville, *Sequential analysis: Hypothesis testing and changepoint detection*. CRC Press, 2014.

[133] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo method*. John Wiley & Sons, 2016, vol. 10.

[134] A. David, K. G. Larsen, A. Legay, M. Mikučionis, D. B. Poulsen, J. Van Vliet, and Z. Wang, "Statistical model checking for networks of priced timed automata," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2011, pp. 80–96.

[135] G. Behrmann, K. G. Larsen, and J. I. Rasmussen, "Priced timed automata: Algorithms and applications," in *International Symposium on Formal Methods for Components and Objects*. Springer, 2004, pp. 162–182.

[136] P. E. Bulychev, A. David, K. G. Larsen, A. Legay, G. Li, D. B. Poulsen, and A. Stainer, "Monitor-based statistical model checking for weighted metric temporal logic." in *LPAR*, vol. 7180. Springer, 2012, pp. 168–182.

[137] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American statistical association*, vol. 58, no. 301, pp. 13–30, 1963.

[138] A. Wald, "Sequential tests of statistical hypotheses," *The annals of mathematical statistics*, vol. 16, no. 2, pp. 117–186, 1945.

[139] D. St-Onge, V. S. Varadharajan, G. Li, I. Svogor, and G. Beltrame, "Ros and buzz: consensus-based behaviors for heterogeneous teams," *arXiv preprint arXiv:1710.08843*, 2017.

[140] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks," *nature*, vol. 393, no. 6684, p. 440, 1998.

[141] M. Otte, M. Kuhlman, and D. Sofge, "Multi-robot task allocation with auctions in harsh communication environments," in *Multi-Robot and Multi-Agent Systems (MRS), 2017 International Symposium on*. IEEE, 2017, pp. 32–39.

[142] C. Pinciroli, A. Lee-Brown, and G. Beltrame, "Buzz: An extensible programming language for self-organizing heterogeneous robot swarms," *arXiv preprint arXiv:1507.05946*, 2015.

APPENDIX A SUPPLEMANTARY MATERIALS FOR CHAPTER 4

Description

This appendix depicts the supplementary support of the research work discussed in Chapter 4.

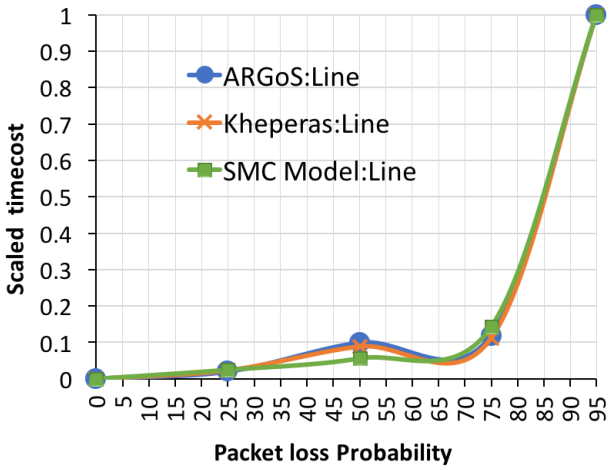
As discussed in Chapter 4, we have used Statistical Model Checking (SMC) to model and assess the robustness of consensus-based behaviors. In a nutshell, our solution models a robot swarm as a network of priced timed automata NPTA. Each robot in the swarm is represented by a single PTA and is able to communicate its state with its neighborhood. The model is weighted by a set of parameters used to assess the impact of the communication quality and robots' defects on the consensus behavior of the swarm. Our model is depicted in the figure below. We have studied the performance of our solution by comparing it with a physics-based simulator (ARGoS) and real-world experiments. We have collected the time to convergence while degrading the communication quality (i.e., increasing packet lost probability). The simulations in the three testing environments are performed on an elect leader scenario implemented in a domain specific language Buzz (see Algorithm 2). The packet loss value is selected from [0%; 25%; 50%; 75%; 95%] and the swarm size is chosen from [5,10].

In ARGoS, we have conducted 500 simulations on cluster, scale free and line swarms with different packet loss probabilities, that is 100 simulations for each packet loss probability. The same has been done in the real word experiments where we used a set of of Khepera IV robots connected by a standard 2.4GHz wireless network.

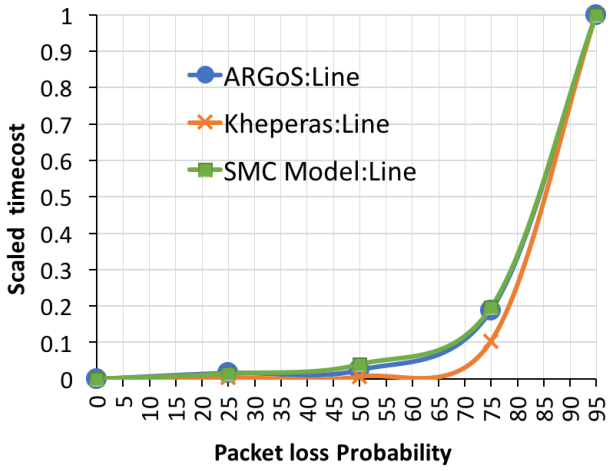
The figures, Fig. A.1, Fig. A.2 and Fig. A.3, depict a comparison between our proposed model (SMC model), ARGoS and real-world experiments. For the 3 studied topologies (Cluster, Scale-free and Line), the figures show that our model exhibits a convergence time similar to what was recorded using ARGoS and Kheperas. These results confirm that our SMC model is representative of the real robot behavior.

Algorithm 2: Virtual Stigmergy: Elect Leader scenario

```
1  VSKEY = 1;
2  vs_value = id;
3  ROBOTS = 10;
4  # The robot with the highest id (10) is elected as a leader;
5  Function init() is
6      # Create a virtual stigmergy;
7      vs = stigmergy.create(VSKEY);
8      # Set onconflict manager ;
9      vs.onconflict(Function (k,l,r) is
10         # Return local value if: ;
11         # - Remote value is smaller than local, OR ;
12         # - Values are equal, robot of remote record is smaller than local one ;
13         if r.data < l.dataor(r.data == l.dataandr.robot < l.robot) then
14             return l ;
15         else
16             # Otherwise return remote value ;
17             return r ;
18         end
19     end
20     # Initialize vstig ;
21     vs.put(VSKEY, vs_value);
22     set_leds(255,0,0);
23 end
24 Function step() is
25     # Get current value ;
26     start_timer();
27     vs_value = vs.get(VSKEY);
28     # If the vs_value corresponds to the highest id ;
29     if vs_value == ROBOTS then
30         stop_timer();
31         log ("I am robot ", id , "my vs_value is", vs_value, "I reached consensus");
32         set_leds(0,255,0);
33     end
34 end
```

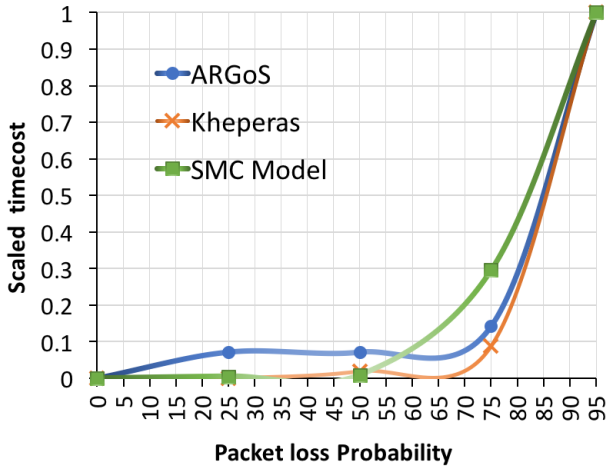


(a) 5-robot line.

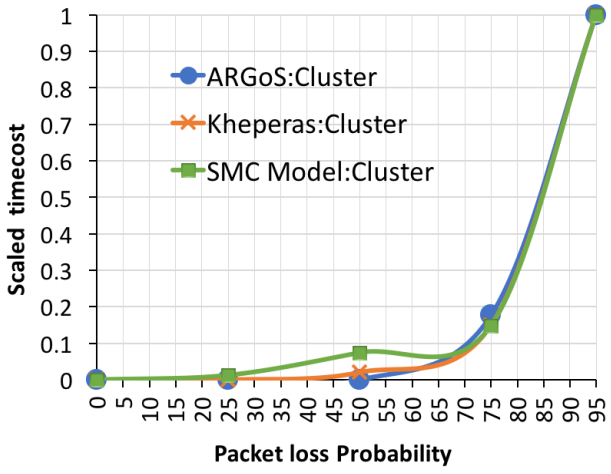


(b) 10-robot line.

Figure A.1 Scaled timecost for an elect leader scenario in a line of robots: Comparison between the SMC Model, ARGoS and real-world experiment.

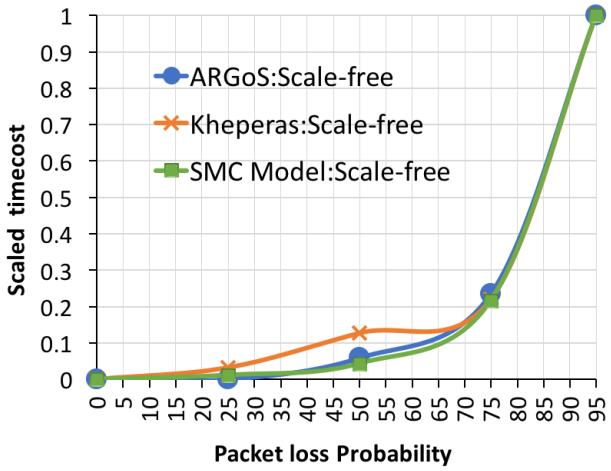


(a) 5-robot cluster.

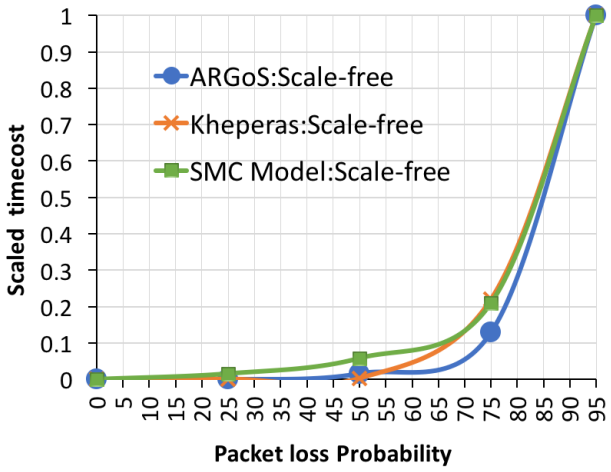


(b) 10-robot cluster.

Figure A.2 Scaled timecost for an elect leader scenario in a cluster of robots: Comparison between the SMC Model, ARGoS and real-world experiment.



(a) 5-robot scale-free.



(b) 10-robot scale-free.

Figure A.3 Scaled timecost for an elect leader scenario by scale-free-connected robots: Comparison between the SMC Model, ARGoS and real-world experiment.