Wrocław University of Science and Technology
Faculty of Electronics, Photonics and Microsystems

FIELD OF STUDY: Control Engineering and Robotics
SPECIALIZATION: Embedded Robotics (AER)

MASTER THESIS

TITLE OF THESIS:
Models and algorithms of supervisory control for multiple mobile robot systems

AUTHOR:
Jakub Kozłowicz

SUPERVISOR:
dr hab. inż. Elżbieta Roszkowska, Prof. PWr

WROCŁAW 2024

Summary

This thesis presents the development of models and algorithms for the supervisory control of multiple mobile robot systems (MMRS). The goal is to ensure efficient and safe navigation of autonomous robots in complex environments, focusing on collision and deadlock avoidance. A hierarchical control system integrates discrete event systems (DES) and continuous-time systems (CTS) to manage both high-level coordination and low-level task execution. Petri nets are used to model coordination layer, and various path discretization methods are evaluated for improving motion efficiency and collision avoidance. Experiments using the Robot Operating System 2 (ROS2) and TurtleBot 2 robots demonstrate significant improvements in multi-robot coordination. This work advances the field of robotics by offering robust solutions for the control of multi-mobile robot systems.

Streszczenie

Niniejsza praca dotyczy opracowania modeli i algorytmów sterowania nadrzędnego dla systemów z wieloma mobilnymi robotami (MMRS). Celem jest zapewnienie efektywnej i bezpiecznej nawigacji autonomicznych robotów w złożonych środowiskach, ze szczególnym uwzględnieniem unikania kolizji i zakleszczeń. Hierarchiczny system sterowania integruje dyskretne systemy zdarzeń (DES) oraz systemy ciągłe (CTS) do zarządzania zarówno wysokopoziomową koordynacją, jak i wykonaniem zadań na niskim poziomie. Sieci Petriego są wykorzystywane do modelowania warstwy koordynującej, a różne metody dyskretyzacji ścieżek są oceniane pod kątem poprawy efektywności ruchu i unikania kolizji. Eksperymenty z wykorzystaniem Robot Operating System 2 (ROS2) oraz robotów TurtleBot 2 wykazują znaczne usprawnienia w koordynacji wielu robotów. Praca ta wnosi istotny wkład w dziedzinę robotyki, oferując solidne rozwiązania dla sterowania nadrzędnego nad systemami wielomobilnymi robotami.

Keywords: robotics, supervisory control, multi-robot systems, path planning, collision avoidance, Petri nets, ROS2

Słowa kluczowe: robotyka, sterowanie nadrzędne, systemy wielorobotowe, planowanie ścieżek, unikanie kolizji, sieci Petriego, ROS2

Contents

Introduction

The significant problems we face cannot be solved at the same level of thinking we were at when we created them.

Albert Einstein

Since the early days of human history, our quest to explore, understand, and replicate the wonders of nature has driven us forward. This relentless curiosity, which has spanned centuries and continents, has been the foundation of monumental discoveries in science and technology. As we transitioned from studying the stars to launching spacecraft, from crafting simple tools to developing complex machines, our progress has been marked by a continual synthesis of knowledge across disciplines. One of the most profound areas of modern exploration is robotics, where the convergence of engineering, artificial intelligence, and control systems is creating unprecedented capabilities [SK16, DJ10, Cor17].

Robotics, a field once confined to the realms of science fiction, has now become a vital area of research and development [SK16, Ark98]. This transformation is reflected in the diversity and complexity of tasks that robots perform today, ranging from surgical procedures and space exploration to disaster response and industrial automation. The term "robot," has evolved significantly from its literary origins to become synonymous with autonomous and intelligent systems that interact with the physical world [TBF05, Par08].

Central to the advancement of robotics is the concept of control systems, which are fundamental to the operation and coordination of robotic entities. Control systems enable robots to perform precise movements, maintain stability, and adapt to dynamic environments [Cor17, YSSA19]. In particular, the supervisory control of multiple robots, which involves the supervision and coordination of the activities of several robots simultaneously, presents a unique set of challenges and opportunities [LaV06, FN87]. This involves intricate processes such as task allocation, path planning, and collision avoidance, all of which must be managed to ensure the effective and safe operation of robots in various settings [JW98, ACF+98].

The focus of this thesis is on the development of models and algorithms for the supervisory control of multiple mobile robots [KFS05]. These robots are tasked with navigating complex environments, which may include obstacles, dynamic elements, and other robots. The objective is to create a control framework that can handle these complexities, ensuring that robots operate efficiently and effectively while avoiding collisions and deadlocks [Bro91]. This involves integrating discrete and continuous control systems to manage high-level coordination and low-level execution of robotic tasks [BR07, MG10].

Scope of the Thesis

The purpose of this thesis is to explore and implement selected models and algorithms underlying the supervisory control of Multiple Mobile Robot Systems (MMRS). The project will investigate the impact of specific control solutions on the behavior of MMRS, providing valuable information on their efficiency and reliability in different operational contexts. The aim is to develop algorithms that facilitate the coordination and control of robots, allowing them to perform tasks in a collaborative and efficient manner. Research focuses on creating a hierarchical control architecture that integrates discrete event systems (DES) and continuous-time systems (CTS), leveraging the strengths of both approaches to achieve optimal control and coordination. The study will examine how various control strategies influence the behavior of multiple robots, focusing on improving their coordination, task allocation, and collision avoidance capabilities.

The **research aspect** involves a comprehensive survey and comparative analysis of different approaches to the discrete representation of concurrent robot motion processes. This includes evaluating control solutions that ensure the correct and efficient operation of MMRS.The study will analyze various control architectures and algorithms to identify their strengths and limitations in handling the complexities of multirobot coordination and control.

The **engineering aspect** of this thesis focuses on the development of a software framework for the MMRS supervisor and mobile robots navigation. This includes designing and implementing the supervisor code, followed by a series of experiments to evaluate the behavior of the MMRS under different control models and algorithms. The experiments aim to evaluate the performance and reliability of the proposed solutions in real-world scenarios, providing empirical data to support the theoretical findings.

By addressing these aspects, this thesis aims to contribute to the field of robotics by providing innovative solutions for MMRS supervisory control. The research findings and developed software tools will enhance the capabilities and reliability of multi-robot systems, enabling them to perform complex tasks in dynamic and unpredictable environments.

Chapter 1

Multi-robot Systems

1.1 Introduction to Multi-robot Systems

Multi-robot systems (MRS) are made up of multiple mobile or stationary autonomous robots that collaborate in distributed way. These systems stand out for their smooth cooperation, which allows them to complete more complicated tasks than one robot could manage on its own. The allocation of tasks among several robots greatly increases work completion times and resource utilization. In addition, the inherent redundancy of MRS improves the fault tolerance and reliability of the system. Multi-Robot systems navigate and carry out missions in both static and dynamic surroundings, frequently in a chaotic and unpredictable condition, by using their aggregate capabilities. This cooperative nature allows adaptability in different operational environments, which makes MRS particularly valuable in situations characterized by environmental uncertainty or task complexity. In addition, MRS facilitates scalability. With an increase in the number of robots, the system can solve more complex or more complex tasks without increasing the proportion of complexity. The modularity of the MRS allows you to easily integrate new robots into existing systems, whether they are identical or varied. The synergy of MRS leads to robust solutions in complex situations such as environmental monitoring, disaster management, and large-scale industrial processes [FHSS20, FIN04, ZHL+18].

1.2 Types of Multi-robot Systems

Multi-robot systems are typically divided into two main types: homogeneous and heterogeneous systems. These systems have been categorized according to the robots that make up them and their similarities or differences in capability [RAT19].

Homogeneous Systems

Each robot in a homogeneous multi-robot system has the same hardware and software features. Since all robots have the same capabilities and limitations, plan, control, and coordination strategies are disentangled, empowering sending and organizing to be more proficient. The most points of interest of a homogeneous framework are that they are simple to operate and keep up since they can utilize the same components and program upgrades anywhere. When employments have great consistency and repetition, such as computerized stockrooms or gathering lines, homogeneous frameworks work well.

However, the flexibility and adaptability of the uniform system are limited. Since all robots are the same, the system may not be able to handle tasks that require various skills. For example, in conditions such as search and rescue missions, a homogeneous system may not function as well as a heterogeneous system. It does not have specialized robots to perform multiple tasks simultaneously, such as debris removal and overhead monitoring. Despite these disadvantages, homogeneous systems are a reasonable option for some applications due to their low cost and simple implementation [SV00, FIN04].

Heterogeneous Systems

Heterogeneous multi-robot systems include robots that are adapted to perform specific tasks in a group and have different capabilities. Due to their differences, team robots are better equipped to deal with a wider variety of tasks since they can individually play their strengths in challenging and changing environments.For illustration, a heterogeneous framework might comprise of ground robots for physical or transport operations and discuss rambles for surveillance, each of which would move forward the capabilities of the other.

The challenge with heterogeneous systems is to make it more difficult to coordinate and facilitate various kinds of robot. In order to guarantee the successful cooperation and smooth communication between the various robots, advanced algorithms and cutting-edge communication protocols are required. These systems often use a distributed control topology, where each robot contributes to the overall goal of the mission, making decisions based on local input. The improved resilience and adaptability of the system allow the plan to deal with unexpected environmental changes and dynamic conditions [SV00, TME14].

Comparison

Both systems have been broadly considered and created, with applications extending from look and protect operations to industrial automation. Homogeneous systems tend to win in situations where tasks are repeated and well-characterized, such as in manufacturing. The consistency of these systems allows for effective large-scale arrangements with the least complexity in control and coordination.

On the other hand, heterogeneous systems are often preferred in dynamic scenarios such as disaster response, where adaptation and specialty are crucial. The various capabilities of heterogeneous robots allow them to tackle various tasks simultaneously, thus improving the overall effectiveness of the mission.

The choice between homogeneous and heterogeneous systems depends on the specific requirements and limitations of the application. Although homogeneous systems offer simplicity and cost-effectiveness, heterogeneous systems offer flexibility and adaptability to more complex and unpredictable environments. Progress in robotics and artificial intelligence continues to blur the boundaries between these categories and enables the development of hybrid systems that combine the best features of the two [BSP23].

1.3 Operating Environments

Multi-Robot systems operate in a variety of environments, including structured and unstructured settings, each presenting unique challenges and opportunities. The choice of environment

significantly impacts the design, control and performance of multi-robot systems, as different tasks and missions require specific capabilities and adaptations [BQ12].

Structured Environments

Structured environments, such as manufacturing floors, are characterized by their defined and predictable nature. These environments are typically well-mapped and have fixed pathways and stations, allowing robots to follow predetermined routes and schedules. The controlled nature of these environments minimizes uncertainties and obstacles, making task execution more efficient.

In structured environments, multi-robot systems can rely heavily on centralized control systems and precise coordination. For example, in an automated warehouse, robots can efficiently manage inventory by following fixed paths to retrieve and store items, reducing human labor and increasing operational speed. The predictability of these settings also simplifies the deployment of homogeneous robot teams, which can be easily managed and maintained due to their uniformity.

Moreover, structured environments benefit from advanced scheduling algorithms that optimize workflow, ensuring that robots perform their tasks in the most efficient sequence and timing. This leads to increased productivity and reduced operational costs. Robotics in structured settings often integrates seamlessly with existing industrial infrastructure, facilitating a smooth transition to automation. However, the rigidity of these environments can limit the flexibility of robots to handle unexpected changes or new tasks without significant reprogramming [Mur04, WRC23].

Unstructured environments

Unstructured environments, such as disaster sites or extraterrestrial surfaces, are characterized by unpredictability and hazards. In these settings, robots must navigate uneven terrain, debris, and unknown obstacles, often requiring real-time decision-making and adaptability. Advanced sensing and autonomous navigation technologies are critical for success in these challenging conditions.

In unstructured environments, the reliance on heterogeneous multi-robot systems becomes more pronounced. Each robot can be equipped with specialized sensors and tools to meet specific challenges. The collaborative effort of diverse robots enhances the overall success of the mission, leveraging the unique capabilities of each type.

Autonomous navigation in unstructured environments requires sophisticated algorithms for path planning, obstacle avoidance, and environment mapping. Robots must be capable of interpreting sensor data on the fly and adjusting their actions accordingly. Machine learning techniques are often used to improve robots' ability to recognize and respond to new situations. Furthermore, robust communication networks are essential to ensure seamless information exchange between robots and their control centers, particularly in remote or hazardous locations where human intervention is limited.

The flexibility of multi-robot systems in unstructured environments allows them to adapt to rapidly changing conditions, making them indispensable in scenarios where human presence is risky or impossible. However, the complexity of these environments requires extensive testing and validation to ensure the reliability and safety of robotic systems. Innovations in materials science, energy storage, and artificial intelligence continue to push the boundaries of what multi-robot systems can achieve in these demanding settings [KKB10, Mur04, WRC23].

1.4 Features and Challenges of Multi-robot Systems

Coordination

Effective task allocation and synchronization between robots are crucial for efficiency and effectiveness. Algorithms must be developed to ensure that robots can dynamically assign tasks among themselves and reallocate resources as needed to handle changing environments and mission requirements. Coordination mechanisms frequently incorporate a combination of centralized and decentralized approaches. Centralized coordination can optimize global goals, but may suffer from single point of failure and scalability issues. In contrast, decentralized coordination improves robustness and scalability but requires sophisticated local decision-making and communication protocols to ensure coherent group behavior [GM04].

Decentralized coordination can involve behavior-based approaches inspired by natural phenomena, such as flocking, where robots follow simple rules to maintain formation and avoid collisions. For example, Reynolds' flocking rules, namely flock centering, obstacle avoidance, and velocity matching, have been adapted for robotic applications, allowing groups of robots to move cohesively while avoiding obstacles and dynamically adjusting their paths [GM12]. Furthermore, hybrid approaches that combine centralized and decentralized strategies can leverage the benefits of both, providing a robust framework for multi-robot coordination [DMG15].

Communication

Reliability in dynamic or hostile environments is essential for coordination. This involves developing robust communication protocols that can handle interference, bandwidth limitations, and the need for secure data transmission. Multi-hop communication, where messages are relayed through intermediate robots to extend the communication range, and the use of communication relays is often employed to maintain connectivity. Furthermore, adaptive communication strategies that adjust transmission power and frequency based on environmental conditions can enhance robustness. Ensuring low-latency and high-throughput communication is critical, especially in time-sensitive applications like search and rescue operations.

Several approaches have been proposed to address the communication challenges in multi-robot systems. One such approach is the use of ad hoc mobile networks, which facilitate flexible and dynamic communication among robots. These networks employ various routing protocols to effectively manage data transmission. For example, the Cluster Head Gateway Switch Routing (CGSR) protocol organizes robots into clusters, with a cluster head responsible for communication within the cluster and with other clusters. This hierarchical approach helps manage the communication load and extend the reach of the network [DMG15].

Moreover, the integration of cloud computing with multi-robot systems offers new avenues for enhancing communication and coordination. Cloud-enabled architectures allow robots to offload computationally intensive tasks to remote servers, reducing the on-board processing load and improving overall system efficiency. This setup also facilitates real-time data sharing and collaborative decision-making among robots, using the computational power and storage capabilities of the cloud [DMG15].

Scalability

Scalability in multi-robot systems is essential for accommodating increasing numbers of robots without significant performance degradation. Scalability in multi-robot systems is largely influ-

enced by the design of the algorithms used for coordination and task allocation. Algorithms should be capable of maintaining performance as the number of robots increases. For example, the study by Portugal and Rocha evaluated five different patrolling algorithms (Conscientious Reactive, Heuristic Conscientious Reactive, Heuristic Pathfinder Conscientious Cognitive, Cyclic Algorithm for Generic Graphs and Multilevel Subgraph Patrolling) in various environments to determine their scalability and performance. The study found that offline planning strategies such as MSP (Multilevel Subgraph Patrolling) and CGG (Cyclic Algorithm for Generic Graphs) generally perform better in weakly connected environments with larger teams, whereas reactive algorithms such as CR (Conscientious Reactive) and HCR (Heuristic Conscientious Reactive) perform better in strongly connected environments with smaller teams [PR13].

Scalability also involves ensuring that all tasks can be scheduled and completed within their deadlines, even as the number of robots increases. Real-time schedulability analysis is essential for this purpose. [SLGR03] propose a heuristic algorithm for real-time schedulability analysis that takes into account communication costs and processor workloads. This approach helps to predict the scalability of the system and design task models that can accommodate additional robots without compromising performance [PR13].

Reliability

Ensuring reliability in multi-robot systems also involves developing robust control algorithms. According to a study by [ZHL+18], a distributed approach to robust control can minimize the impact of individual robot failures on the overall system. The robust control algorithms proposed ensure that the failure of a robot affects only those directly interacting with it, preventing a cascade of failures throughout the system. This is achieved by dividing robots into reliable and unreliable categories and using distributed control strategies to manage their interactions and mitigate the effects of failures [ZHL+18].

Another important aspect of reliability is the detection and recovery from failures. Implementing real-time health monitoring and diagnostic systems can help identify potential failures before they occur. Techniques such as redundancy, where multiple robots perform the same task to ensure that failure of one does not compromise the mission, and fault-tolerant designs that allow robots to detect and recover from failures autonomously, are critical to maintaining reliability in multi-robot systems [ZHL+18].

In addition, task reallocation strategies play a vital role in maintaining reliability. When a robot fails, the system must be able to dynamically reallocate its tasks to other robots to ensure mission continuity. This requires sophisticated algorithms that can quickly and efficiently redistribute tasks without causing significant delays or disruptions. The use of machine learning and artificial intelligence can further enhance the reliability of task reallocation by allowing the system to learn from past failures and improve its decision-making processes over time [ZHL+18].

In addition, robust communication protocols are essential for reliability. In environments where communication can be intermittent or unreliable, ensuring that robots can still coordinate effectively is crucial. Techniques such as multi-hop communication, where messages are relayed through multiple robots to extend the communication range, and adaptive communication strategies that adjust parameters based on current conditions, can help maintain reliable communication links between robots, even in challenging environments [ZHL+18].

Collision and Deadlock Avoidance

Robots must be able to detect and avoid collisions with each other and with obstacles in real-time. This requires sophisticated sensing systems, such as LiDAR, radar, and computer vision, and real-time processing capabilities to interpret sensor data and make quick decisions. Predictive modeling, where robots anticipate the future positions of obstacles and other robots, and machine learning techniques, which allow robots to learn and improve their collision avoidance strategies from experience, are often employed to enhance performance.

Recent approaches to collision avoidance in multi-robot systems have emphasized the use of distributed algorithms. [ZHL+18] propose a distributed real-time algorithm for the avoidance of collisions and deadlock. In this method, each robot autonomously checks its succeeding states and stops if a collision or deadlock is predicted. This algorithm ensures that no more than one robot occupies a given collision zone at a time, thereby preventing collisions and improving the overall performance of the system [ZHLD17].

The algorithm models the motion of each robot as a labeled transition system, where each robot can autonomously execute the algorithm to avoid collisions and deadlocks. When a robot detects that its next move could lead to a collision or a deadlock, it stops and waits for the other robot to move first. This decision-making process is managed locally by each robot, allowing the system to be scalable and robust [ZHLD17].

In addition to avoiding collisions, the algorithm addresses deadlocks by introducing negotiation strategies. When multiple robots need to move to the same location, they negotiate to determine which robot moves first, thus avoiding decision deadlocks and improving the efficiency of the system. If a deadlock is detected, the robots involved negotiate to resolve the deadlock, ensuring that at least one robot can move and thus breaking the deadlock cycle [ZHLD17].

The distributed approach also incorporates advanced collision avoidance strategies that combine high-level discrete control with low-level continuous feedback control. This dual-layered control mechanism allows for more effective planning and execution of collision avoidance maneuvers, providing a comprehensive solution that seamlessly integrates both phases [ZHLD17].

Ensuring safe and efficient navigation in dynamic environments is a critical aspect of multi-robot system design. Continuous advancements in sensor technology and computational algorithms are essential to maintaining high performance and reliability in avoiding collisions and deadlocks. Using these advancements, multi-robot systems can achieve greater autonomy and robustness in various applications, from search and rescue missions to industrial automation [ZHLD17].

1.5 Multi-mobile Robot Systems

Multi-mobile robot systems (MMRS) refer to a subclass of multi-robot systems characterized by the mobility of the robots involved. These systems consist of multiple autonomous mobile robots that collaborate to perform tasks in various environments, representing a highly adaptive and scalable approach to the execution of complex tasks. MMRS leverages advanced coordination algorithms to manage dynamic task allocation and real-time synchronization, crucial for navigating unpredictable environments. Robust communication protocols ensure seamless information exchange, allowing coordinated efforts among robots. Scalability is maintained through algorithms designed to handle an increasing number of robots without significant performance degradation, utilizing decentralized control and real-time schedulability analysis. Reliability is achieved through fault-tolerant designs, redundancy, real-time health monitoring, and diagnostic systems, which detect and address potential failures autonomously. The integration of machine learning and AI

allows MMRS to learn from past experiences, continually improving reliability and performance. These systems exemplify the potential of collaborative robotics in improving efficiency, adaptability, and robustness, effectively tackling applications ranging from industrial automation to disaster response and environmental monitoring [GM12, ZHLD17].

Chapter 2

Robot Control

2.1 Introduction to Robot Control

Robot control is a crucial aspect of modern robotics, focusing on the design and implementation of controllers to achieve desired behaviors in robotic systems. Initially, the field of robotics was limited by high computational costs, inadequate sensors, and a fundamental lack of understanding of control systems, restricting applications to simple tasks like material handling and welding. However, advances in technology and control theory have expanded the capabilities of robotic systems, enabling them to perform complex tasks such as planetary exploration and autonomous navigation. A robot controller, a vital component, defines the accuracy and repeatability of a robot by modifying its behavior through computational algorithms and actuation mechanisms. Effective control systems can be classified into open-loop and closed-loop types, with the latter incorporating feedback to improve stability and performance. As robotics continues to evolve, the integration of more intelligent control strategies, such as supervised control for multi-mobile robot systems and discrete control methods, becomes essential to meet the increasing complexity and demands of modern applications. The complexity of these systems arises from the need to manage and coordinate multiple robots simultaneously, each performing different tasks while avoiding collisions and ensuring efficient operation [Bad15].

2.2 Types of Control Architectures

Control architectures define the structure and organization of the control system and determine how robots interact with each other and the environment. These architectures can be classified into several categories, each with its advantages and disadvantages.

Centralized control involves a single supervisor who oversees the entire system, makes decisions, and coordinates robot activities. This approach is simple and efficient for small systems, but it can become a bottleneck as the system scales, leading to increased computational complexity and communication overhead. Centralized control architectures are suitable for applications that require tight coordination and synchronization between robots, such as formation control and cooperative manipulation [HAPG21, RJ21, RMCJ23]. The central supervisor's ability to have a global view of the system allows optimal decision-making, but this central point of failure can significantly affect the entire system if it fails.

Decentralized control distributes decision making among robots, allowing them to coordinate their actions independently based on local information. This approach is more scalable and robust

than centralized control, but can lead to conflicts and inefficiencies if robots do not communicate effectively [HAPG21, RJ21, RMCJ23]. Decentralized control architectures are suitable for applications that require flexibility and adaptability, such as search and rescue missions and surveillance. The lack of a single point of failure enhances system robustness, but achieving global system objectives can be challenging without effective inter-robot communication.

Distributed control combines elements of centralized and decentralized control, with multiple supervisors working together to manage different aspects of the system. This approach is flexible and scalable, making it suitable for large, complex systems with diverse requirements. Distributed control architectures are suitable for applications that require fault tolerance and redundancy, such as multi-robot exploration and mapping [RJ21, RMCJ23]. Using multiple supervisors, the system can balance the benefits of both centralized decision-making and decentralized autonomy, thus improving overall system performance and reliability.

Hierarchical control architectures organize the control system into multiple levels of abstraction, each level responsible for a different aspect of the system. This approach enables supervisors to manage the system at multiple levels, ensuring that the robots work seamlessly to achieve their objectives [HAPG21, RMCJ23]. Hierarchical control architectures are suitable for applications that require complex coordination and cooperation between robots, such as multi-robot manipulation and assembly [HAPG21, RMCJ23]. By hierarchically structuring the control system, it is possible to manage detailed local interactions at lower levels while guiding the overall behavior of the system at higher levels, thus optimizing both local and global performance.

By selecting the appropriate control architecture for a given application, supervisors can ensure that the robots work together effectively to achieve their objectives. This decision is influenced by factors such as the size and complexity of the system, the level of coordination required, and the availability of communication and computational resources. Using the advantages of different control architectures, supervisors can design robust, efficient, and scalable control systems that enable robots to perform complex tasks in a wide range of applications.

2.3 Supervisory Control in Multi-robot Systems

Supervisory control in multi-robot systems involves overseeing and directing robot activities at a high level, while robots handle low-level execution autonomously. This approach ensures that robots can perform complex tasks efficiently and safely. Supervisory control in multi-robot systems includes mission planning, task allocation, and monitoring. The supervisor sets the overall mission objectives and constraints, while algorithms help break these objectives down into specific tasks that robots can execute. The supervisor also monitors the system's performance and intervenes when necessary to adjust strategies or respond to unexpected situations. A multilevel hierarchical control system is commonly used in multi-robot systems, consisting of three levels of control: the top level being a supervisor based on a discrete representation of the Multiple Mobile Robot System (MMRS), an intermediate level supervising the execution of robot motion on individual stages, and the lowest level responsible for actual motion execution. The top level supervisor ensures collision- and deadlock-free movement by centrally controlling changes in robot stages, viewing robot motion processes as sequences of stages [RMCJ23].

2.3.1 Hybrid Control Systems

Hybrid control systems integrate discrete event systems (DES) and continuous-time systems (CTS) to leverage the strengths of both. DESs are adept at handling discrete event-driven behav-

iors, such as task initiation and completion. They are used to model discrete and logical aspects of robotic systems, such as task transitions, synchronization, and decision points. They provide a structured way to handle sequences of events and state changes, essential for coordinating complex multi-robot tasks. On the other hand, CTS manages continuous dynamic behaviors such as trajectory tracking, velocity control, dynamic response, and speed control. They use differential equations and control theory to model and control the physical movements of robots, ensuring smooth and precise execution of tasks [RMCJ23].

To ensure efficient and correct operation of the MMRS, the supervisory control uses a DES model to represent concurrent robot movements along specified paths. This model employs the Petri net formalism to ensure the required logic of MMRS operations, such as avoiding collision and deadlock, through a set of control procedures that adjust robot motions according to supervisor decisions. The low-level robot control, based on CTS, ensures that the robots follow the desired paths and velocity profiles accurately [RMCJ23].

By combining these approaches, hybrid systems can provide a comprehensive framework for robot control, allowing precise motion execution alongside robust event handling. This modular construction of the hybrid control system also facilitates modifications and experimental examinations to optimize MMRS performance under various operational conditions [RMCJ23].

2.3.2 Supervisory Control Models

Supervisory control models for multi-mobile robot systems utilize discrete event systems (DES) to manage the coordination and control of multiple robots. The primary model used in this context is the Petri net formalism, which provides a robust framework for representing concurrent processes and their interactions. Petri nets are particularly useful for modeling the synchronization and conflict resolution required in multi-robot systems. They allow the representation of complex dependencies and constraints, ensuring that robots operate without collisions and deadlocks [RJ21].

A typical Petri net model for MMRS includes places representing different states or locations of robots, transitions that indicate state changes or movements, and tokens that mark the current state of the system. The control logic is embedded in the structure of the Petri net, dictating the permissible sequences of actions based on the system's current state. This method ensures that all robots can complete their tasks efficiently while complying with the necessary safety and operational constraints [CL10].

Another widely used model for supervisory control in multi-robot systems is the Finite State Machine (FSM). FSMs are computational models that represent the system's states and the transitions between these states. In the context of multi-mobile robot systems, FSMs can be employed to model and control the behaviors of individual robots as well as their interactions. Each state in an FSM represents a specific behavior or mode of operation of a robot, and transitions between states are triggered by events or conditions in the environment [CL10].

FSMs are particularly advantageous because of their simplicity and ease of implementation. They provide a clear and intuitive way to design control logic by defining explicit states and transitions. This makes it easier to debug and verify the system's behavior. Moreover, FSMs can be extended to Hierarchical Finite State Machines (HFSMs), which introduce hierarchy into the state machine, allowing more complex behaviors to be managed in a modular and scalable manner [CL10].

Supervisory control models are essential for managing the interactions between robots in multi-robot systems, ensuring they work together effectively to achieve common goals. These models can be classified into centralized, decentralized, and distributed control architectures, each with its

unique advantages and disadvantages. Furthermore, supervisory control models vary on the basis of the level of abstraction they provide. High-level models focus on mission planning and task allocation, whereas low-level models focus on motion planning and execution. By integrating these models, the supervisors can manage the system at multiple levels, ensuring that the robots work seamlessly together to achieve their objectives [RMCJ23].

2.3.3 Algorithms for Supervisory Control

Supervisory control algorithms play a crucial role in managing multi-robot systems, ensuring that the robots can work together effectively to achieve their objectives. These algorithms can be classified into several categories, each with its strengths and weaknesses. The key algorithms used in supervisory control of multi-robot systems include task allocation algorithms, path planning algorithms, and decision-making algorithms.

Task Allocation Algorithms

Multi-robot Task Allocation (MRTA) is a complex problem within the field of multi-robot systems (MRS) that involves optimally assigning a set of robots to a set of tasks to maximize overall system performance under various constraints. This problem is particularly challenging when dealing with heterogeneous robots equipped with different capabilities and required to perform tasks with diverse requirements and constraints. There are two primary approaches to solving the MRTA problem: metaheuristic-based and market-based approaches. Metaheuristic-based approaches, such as genetic algorithms and ant colony optimization, utilize population-based algorithms that leverage multiple agents to search for an optimal solution. These approaches have been shown to be effective in handling complex, heavily constrained applications that involve numerous heterogeneous tasks and robots. For example, genetic algorithms have been applied to group tracking systems and time-extended task allocation scenarios, while ant colony optimization has been used to solve cooperation tasks among multiple robots [KHE15]. Market-based approaches, on the other hand, involve the allocation of tasks through bidding mechanisms, where robots bid for tasks based on their capabilities and costs. This method has proven effective in scenarios where task requirements and robot capabilities need to be matched dynamically. Studies have shown that while metaheuristic approaches often outperform market-based approaches in scalability, both methods are comparable in handling real-world constraints such as time and robot capability-task requirement matching [KHE15].

The MRTA problem can be categorized into different schemes and organizational paradigms. Schemes include single-task (ST) versus multi-task (MT), and single-robot (SR) versus multi-robot (MR), indicating whether robots perform tasks sequentially or simultaneously and whether tasks require one or multiple robots. Organizational paradigms can be centralized, where a central agent allocates tasks, or decentralized, where each robot makes task allocation decisions independently. Centralized approaches, while effective for small-scale and static environments, face scalability issues and potential single points of failure. Decentralized approaches, such as consensus-based auction algorithms and hierarchical market-based methods, offer robustness and flexibility, making them suitable for dynamic and large-scale environments [KHE15].

Path Planning Algorithms

Path planning for multi-mobile robot systems involves determining optimal and collision-free paths in dynamic environments, and it is integral to supervisory control strategies. The hierar-

chical approach to path planning includes three levels: the Supervisory Global Planner (SGP), the Global Planner (GP), and the Local Planner (LP). The SGP operates on a static map, designating waypoints that the GP uses to generate a global path while considering both static and dynamically detected obstacles. The LP further refines this path in real-time to avoid unforeseen obstacles. Various path planning algorithms play a crucial role in this process. Graph-based algorithms such as A*, Dijkstra's, and D* are foundational, with A* being particularly noted for its efficiency in balancing cost and distance to the target [IPS+19]. Sampling-based algorithms, including Rapidly-exploring Random Trees (RRT) and Probabilistic Roadmaps (PRM), provide solutions for complex, high-dimensional spaces by randomly sampling feasible paths [IPS+19]. Optimization-based methods like Mixed-Integer Linear Programming (MILP) and Sequential Quadratic Programming (SQP) offer precise, constraint-based solutions, but are computationally intensive [IPS+19]. The hierarchical supervisory control system ensures that the global path integrates a safe trajectory, continuously updated by the local planner to navigate around dynamic obstacles effectively. This approach has been experimentally validated in scenarios such as industrial environments, where robots must navigate safely among humans and machinery [IPS+19].

Real-time Decision-making Algorithms

Decision-making algorithms in multi-agent systems (MAS) are critical for enabling agents to accomplish complex tasks cooperatively. These algorithms range from Markov decision processes (MDP), game theory, swarm intelligence, to graph-theoretic models, each with its strengths and applications. MDPs are widely used for modeling decision making where the outcomes are stochastic and depend on the current state and action taken, satisfying the Markov property where the future state depends only on the current state and action. This approach is extended to partially observable environments using POMDPs, which account for uncertainties and incomplete information, making them suitable for dynamic and unpredictable settings [RAT18]. Game theory, traditionally used for competitive environments, has found applications in cooperative MAS by enabling agents to model and predict the actions of others, thus optimizing collective outcomes. Bayesian games, a subset of game theory, handle uncertainties in agents' rewards, crucial for strategic decision-making in environments with incomplete information [RAT18]. Swarm intelligence, inspired by social animals, employs simple rules and local interactions among homogeneous agents to achieve global objectives, making it robust and scalable for tasks such as foraging and path planning [RAT18]. Graph-theoretic approaches, such as influence diagrams (IDs), provide a structured framework for decision-making by incorporating actions and utilities into Bayesian networks, allowing the representation of complex dependencies and sequential decision making [RAT18]. These algorithms must also consider factors such as system heterogeneity, computational efficiency, and adaptability to dynamic environments to be effective in real-world applications, including robotics, intelligent transport systems, and smart grids. Despite significant advancements, challenges remain to develop scalable, autonomous, and efficient algorithms that take advantage of the advances of big data and the Internet of Things (IoT) to handle increasingly complex tasks in MAS [RAT18].

2.4 Low-level Robot Control

Low-level robot control involves the execution of specific movements and actions as dictated by higher-level supervisory control. Objectivities are to ensure that robots follow planned routes accurately, maintain desired speeds, and respond dynamically to changes in their environment.

2.4.1 Path-following Algorithms

In the context of low-level robot control, path-following algorithms play a crucial role in ensuring that each robot adheres to a predefined path with minimal deviation. The main goal of these algorithms is to minimize the distance between the robot's current position and its desired path while also controlling the orientation and velocity to achieve smooth motion. Path-following techniques are essential for the effective operation of multiple mobile robot systems (MMRS), particularly in scenarios where precise navigation is required to avoid obstacles and ensure mission success.

Commonly used path-following algorithms include pure pursuit, which involves calculating a target point along the desired path at a fixed look-ahead distance and steering the robot towards this point. This method is simple and effective for smooth paths, but it can be difficult to maneuver with sharp turns and high-speed maneuvers [C+92]. Another widely adopted method is the Stanley controller, which focuses on minimizing the lateral error and heading error to maintain the robot's alignment with the path. This approach is particularly effective for autonomous vehicles operating at various speeds and has been successfully implemented on different robotic platforms [TMD+06].

Advanced path following techniques often incorporate model predictive control (MPC), which uses a dynamic model of the robot to predict its future states and optimize the control inputs over a finite time horizon. This method allows for a more sophisticated handling of constraints and dynamic environments, making it suitable for complex scenarios with moving obstacles [QB03]. Additionally, sliding mode control offers robustness against disturbances and model uncertainties by switching between different control laws based on the robot's state, ensuring reliable path tracking even in challenging conditions [Utk92].

The integration of these algorithms within a hybrid control system, which combines discrete event supervision control with continuous-time motion control, improves the overall performance of MMRS.The supervisory level handles high-level task allocation and coordination, ensuring collision avoidance and mission completion, while the low-level control algorithms manage the real-time execution of path following tasks [RJ21]. This hierarchical approach allows for efficient and scalable control of multiple robots, allowing them to operate concurrently in shared workspaces without interfering with each other's paths [RJ21].

2.4.2 Motion Control Procedures

Motion control algorithms for mobile robots are a critical aspect of low-level robot control, ensuring the precise and reliable execution of movements in dynamic environments. These algorithms typically integrate various control mechanisms to achieve accurate path follow and obstacle avoidance. One widely used approach is differential drive control, which calculates the velocities of the left and right wheels of the robot to achieve the desired trajectory and orientation [BV00]. This method involves reactive control equations that adjust wheel velocities based on the robot's current position and the target configuration, effectively handling both stationary and moving targets through continuous replanning and feedback adjustment [BV00].

Advanced implementations of these algorithms often incorporate predictive models, such as Kalman filters, to estimate the positions of dynamic obstacles and adjust the robot's path accordingly. The use of such models improves the robot's ability to navigate complex environments, as demonstrated in robotic soccer scenarios, where robots must maneuver around opponents and intercept a moving ball [BV00, RJ21]. Additionally, path-following algorithms like pure pursuit and its spline-based modifications are employed to minimize tracking errors and ensure smooth motion along predefined paths. These algorithms calculate the curvature needed to follow the path

accurately, adjusting the robot's velocity and orientation to reduce deviations [RJ21].

In summary, motion control algorithms for mobile robots combine differential drive control, predictive modeling, and advanced path-following techniques to achieve robust and precise navigation in dynamic environments, ensuring effective low-level control essential for various applications [BV00, RJ21, RMCJ23].

Chapter 3

Models and Algorithms of Supervisory Control

The system considered in this thesis is a multi-mobile robot system (MMRS), where multiple homogeneous robots are controlled by a central controller. The robots have the same capabilities and share the same static environment. The environment is a 2D motion space with obstacles and other robots. The robots plan their paths independently based on the environment map and execute them using their motion control algorithms (path following). The objective of the central controller is to manage the movement of multiple robots, ensuring that they can navigate safely without colliding with other robots or coming to a situation where no robot can move forward (deadlocks) [RMCJ23, RJ21].

The system employs a hierarchical and hybrid control architecture (see Fig. 3.1) to manage robot operations, ensuring safe navigation and avoiding collisions and deadlocks. The control system is structured into three distinct levels [RJ21]:

1. **High-Level Supervisory Control:** This level supervises the general movement and coordination of robots across different path segments. It ensures that robots adhere to the mission plan and do not collide or enter into deadlock situations. The supervisory controller uses discrete event systems (DES) to model and manage the sequence of robot movements, ensuring efficient coordination and task execution.
2. **Mid-Level Stage Traverse Control:** Acting as an intermediary, this level bridges the discrete supervisory control of the high level and the control of the low level continuous motion. Manages the transitions between discrete path segments and continuous motion, ensuring that robots move smoothly and effectively from one segment to another.
3. **Low-Level Robot Motion Control:** This level is responsible for direct control of robot movements. It uses continuous-time systems (CTS) to manage the physical dynamics of each robot, ensuring accurate path-following and real-time response to changes in the environment. The robots execute their planned routes using motion control algorithms that adhere to kinematic and dynamic models.

3.1 Path Disctretization

The task of path discretization is to divide the continuous path into discrete segments that can be assigned to individual robots. This process is crucial for coordinating the movement of multiple robots. Each robot can be assigned to one segment at a time. This ensures that the robots

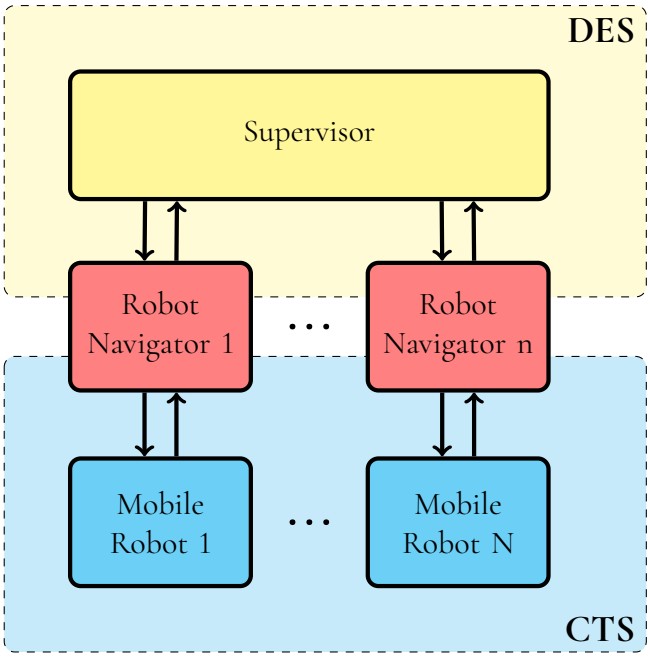


Figure 3.1: Hierarchical system model [RJ21]

can navigate safely without colliding or entering deadlock situations. The discretization process is required to transform the continuous movement of a robot into a discrete process. The path discretization can be performed using various methods, such as equal-length discretization, optimal length discretization, or grid-based discretization. The choice of method depends on the specific requirements of the system and the environment in which the robots operate.

3.1.1 Equal Length Discretization

Equal-length path discretization is the most straightforward method of path discretization, in which the path is divided into segments of uniform length. This method simplifies the assignment of segments to robots, as each segment maintains a consistent length. However, this approach may not always be optimal as it divides the path into equal parts regardless of intersections or crossings with other paths. The algorithm iterates through the path, cumulatively summing the distances between each pair of points and determining whether a segment should be created. When the cumulative distance exceeds a predefined length, a segment is created. This process is repeated until the entire path has been segmented. Figure 3.2 shows an example of discretization of paths using the equal length method.

3.1.2 Optimal Length Discretization

Optimal length path discretization represents a more advanced method that takes into account proximity to other paths when segmenting the path. This approach aims to maximize the lengths of collision-free segments while minimizing the lengths of segments where collisions are possible. The safe distance is defined as the sum of the robots' radiuses and an additional buffer distance to prevent collisions. The algorithm iterates along the path, continuously assessing the distance to neighboring paths and creating segments whenever this distance falls below the predefined safe distance. Figure 3.3 shows an example of the optimal length path discretization. In a situation where the paths do not cross, the optimal length path discretization is equivalent to no discretization.

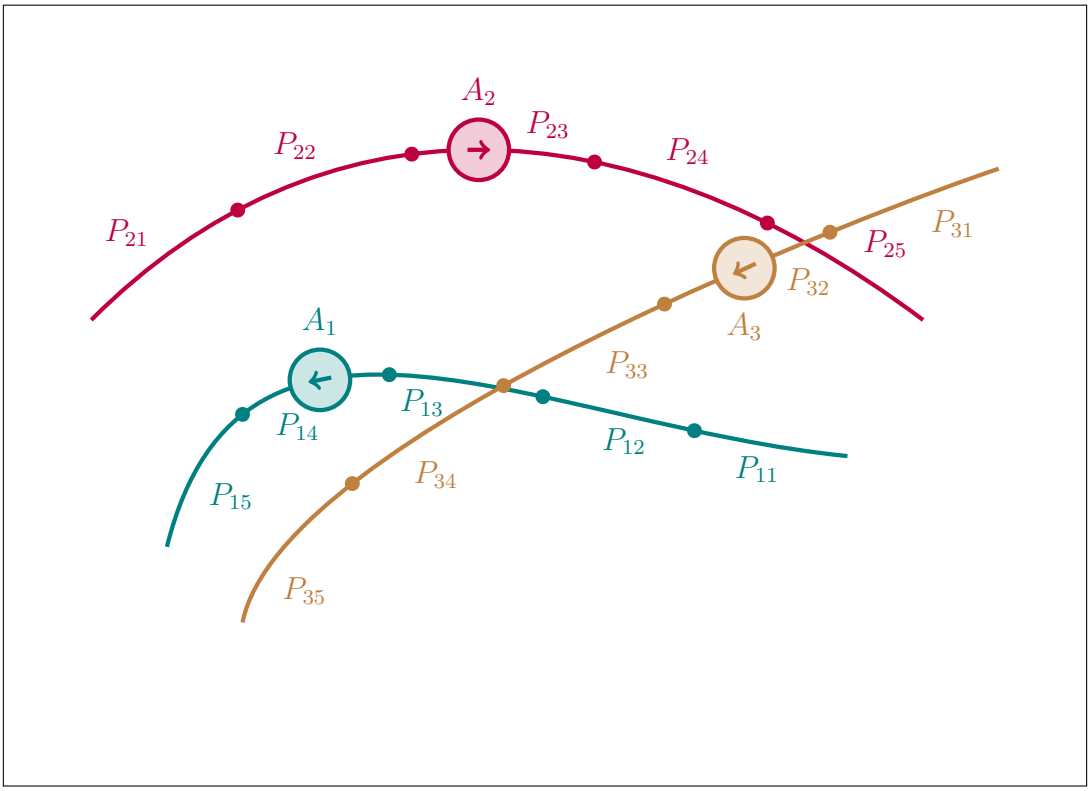


Figure 3.2: Paths for multiple robots divided equally

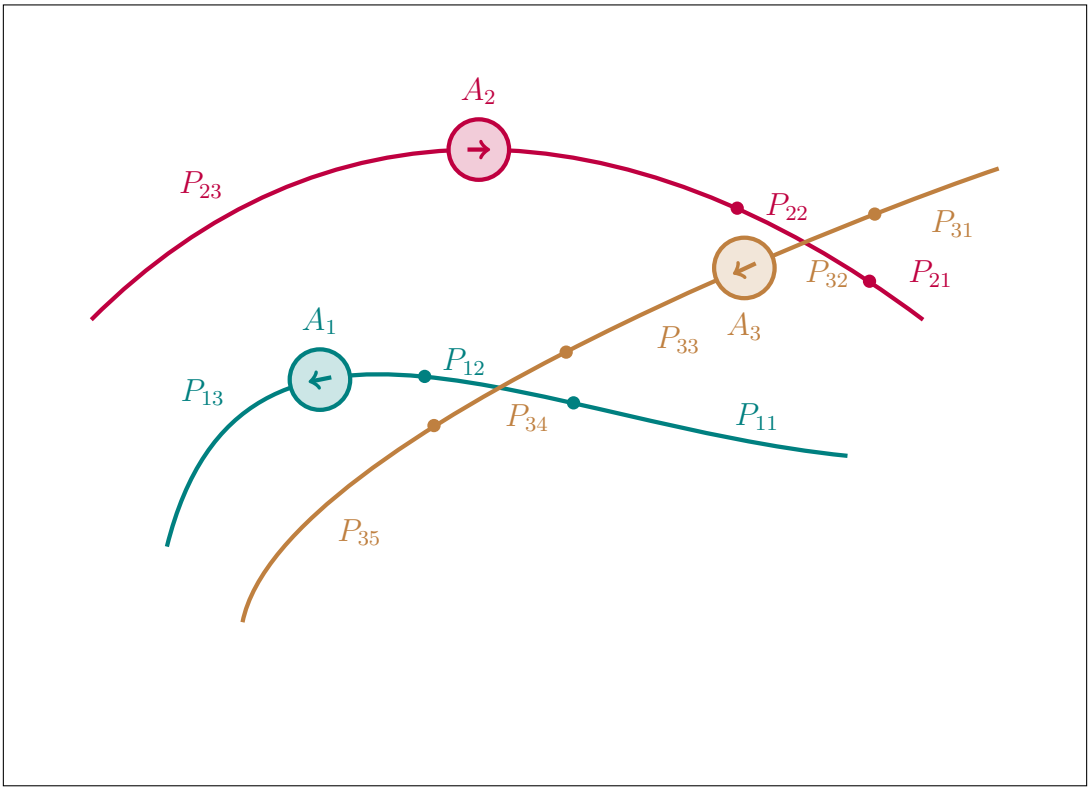


Figure 3.3: Paths for multiple robots divided optimally

3.1.3 Grid Based Discretization

Grid-based path discretization is based on a virtual grid map tailored to the dimensions of the robot. Each cell in this grid is square, with side lengths corresponding to the robot's diameter plus additional buffer. At the junctions of these square cells, circular cells with a radius equal to the robot's radius are placed, accommodating scenarios where the robot occupies four adjacent cells. Additionally, smaller rectangular cells, with a width equivalent to the robot's radius, are located next to the sides of the square cells. This arrangement forms a comprehensive grid map that highlights subcells, indicating whether a robot occupies a single cell, multiple cells, or overlaps between cells. Path discretization follows this virtual grid, creating segments whenever the path intersects these cells. For simplicity of the path division, segments that indicate requirement of three and four adjacent cells are joined together. This reduces the occurrences of short segments in which whole robot will not be contained. The concept of grid-based path discretization is depicted in Figure 3.4. The detailed part outlined by the circle is shown in Figure 3.5.

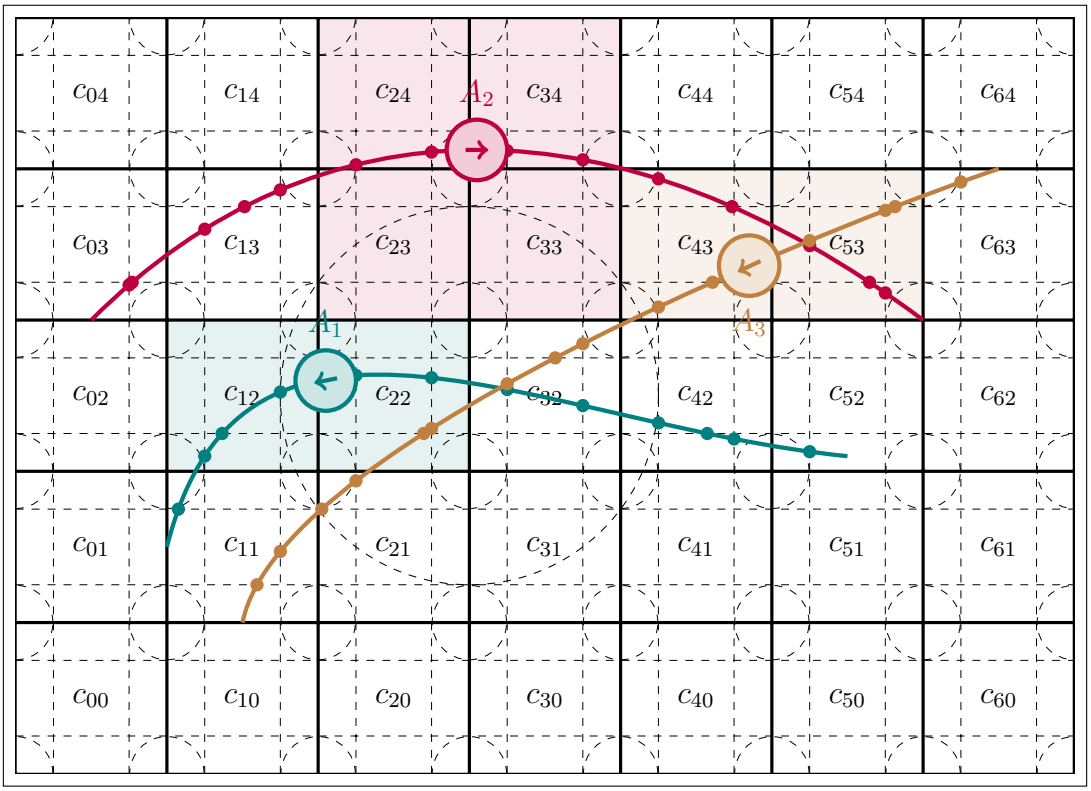


Figure 3.4: Paths for multiple robots divided by grid

3.2 Supervisory Control Model

The supervisory control model for a multi-mobile robot system (MMRS) presented in this thesis leverages the Petri nets formalism to ensure efficient and collision-free coordination among multiple robots. The model is designed to manage the discrete events and states associated with robot movements, while integrating with the continuous control of robot kinematics and dynamics.

Based on [RJ21], the definition of the Place/Transition (P/T) system is presented in Definition 1 along with the definition of the state of the system and the firing rules for transitions in the system

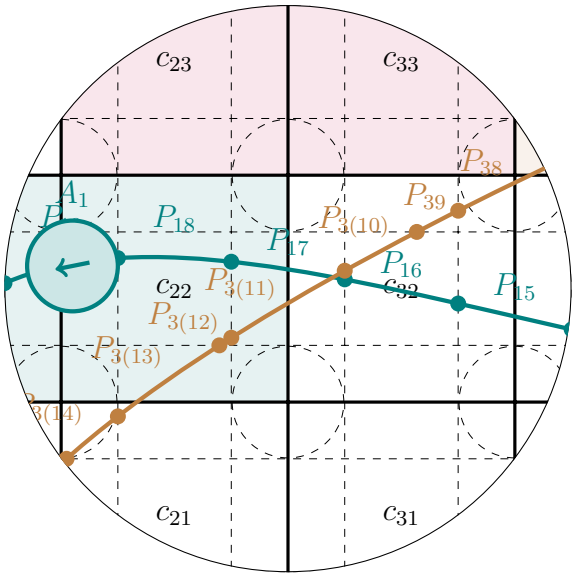


Figure 3.5: Detail of paths for multiple robots divided by grid

in Definition 2. An enabled transition and the marking of the net before and after its firing are shown in Figure 3.6.

Definition 1 *A place/transition system (P/T system) is a triple* $N = (P, T, F, M_0)$ *such that:*

- $(P \cup T, F)$ *is a net where*
  - $P \cup T$, $P \cap T = \emptyset$, *is the node set, that comprises two types of nodes: places* $p \in P$ *and transitions* $t \in T$. *The set* T *is further partitioned into two sets* $T_C \cup T_O$, $T_C \cap T_O = \emptyset$, *of controllable and observable (non-controllable) transitions.*
  - $F \subseteq (P \times T) \cup (T \times P)$ *is the flow relation, that constitutes the arc set of the net.*
- $M_0 : P \to N$ *is the initial state of the system, called the initial marking.*

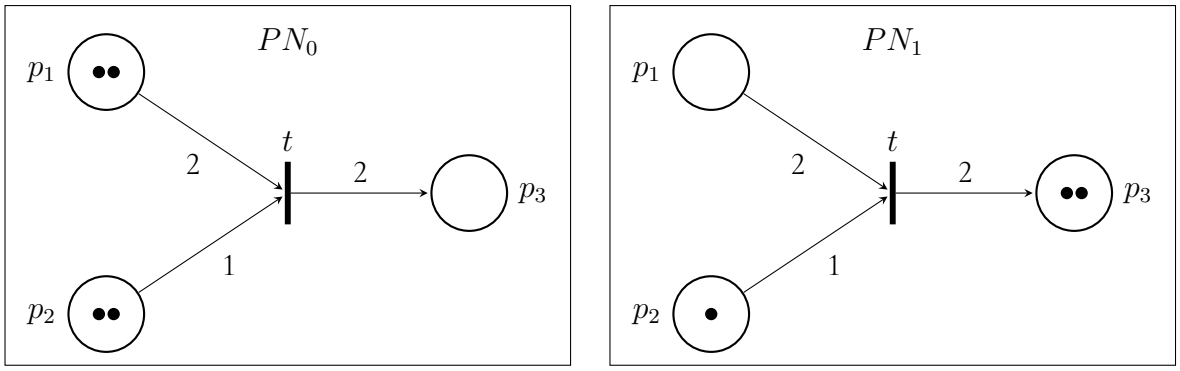
Definition 2 *The state of a P/T system is represented by marking* $M_0 : P \to N$, *that is initially given by* M_0 *and then changes as a result of firing transitions according to the following rules:*

- *only an enabled transition* t *can fire*
- *a transition* t *is enabled in marking* M *if and only if* $\forall p \in {}^\bullet t : M(p) \geq 1$
- *if transition* t *fires in marking* M *then it causes the change of the marking to* M_0 *such that:*

$$M'(p) = \begin{cases} M(p) - 1 & \text{if } p \in {}^\bullet t \backslash t^\bullet \\ M(p) + 1 & \text{if } p \in t^\bullet \backslash {}^\bullet t \\ M(p) & \text{otherwise} \end{cases}$$

where:

- ${}^\bullet t = \{p \in P : (p, t) \in F\}$ *is the set of input places of transition* t,
- $t^\bullet = \{p \in P : (t, p) \in F\}$ *is the set of output places of transition* t.



(a) Petri net with enabled transition t

(b) Marking of the Petri Net resulting from firing transition t

Figure 3.6: Petri net transition

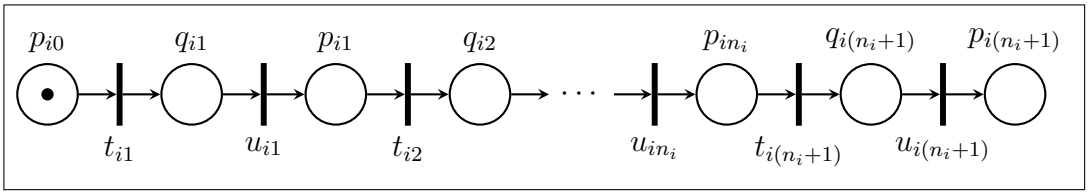


Figure 3.7: Petri net model for path traversal of a single robot [RJ21]

The supervisory controller benefits from the Petri nets formalism to model the discrete events and states associated with robot movements based on the discretization of the paths. The Petri net model for a single robot is shown in Figure 3.7. This diagram illustrates a Place/Transition (P/T) system that models the movement of robot A_i along its designated path p_i, which is divided into n segments. The visual representation includes various components:

- **Places**: Represented by circles (e.g. $p_{i0}, q_{i1}, p_{i1}, q_{i2}, \ldots$), these indicate specific states or conditions of the robot. For example, p_{i0} and $p_{i(n+1)}$ symbolize the position of the robot at the beginning and end of its path, respectively. The places q_{i1} to $q_{i(n+1)}$ mark the robot's entry and exit points for each path segment. The places p_{ij}, for $j = 1, \ldots, n_i$, represent the travel of robot A_i along the different sectors of its path. Similarly, places $q_{i,j}$, for $j = 2, \ldots, n_i$, denote the transitions between consecutive sectors p_{ij} and $p_{i(j+1)}$.
- **Transitions**: Depicted as black bars or short vertical lines (t_{ij}, u_{ij}), these represent the events or actions that change the robot's state.
  - Transitions t_{ij}, for $j = 1, \ldots, (n_i+1)$ are controllable events representing permissions for the robot A_i to enter the segment p_{ij}. For example t_{i1} and $t_{i(n+1)}$ indicate when the robot is ready to move into the path and when it is ready to complete the path, respectively. Transitions t_{ij}, for $j = 2, \ldots, n_i$, occur when the robot is ready to move to the next sector p_{ij} upon completing the current sector $p_{i(j-1)}$.
  - Transitions u_{ij}, for $j = 1, \ldots n_i + 1$ are observable events that occur when the robot fully enters or exits a specific path segment. For example, u_{i1} and $u_{i(n+1)}$ indicate when the robot completely moves into the first segment or out of the path, respectively. Specifically, u_{ij}, for $j = 2, \ldots, n_i$, occur when the robot's disk entirely enters sector p_{ij}, signifying no overlap with the previous sector $p_{i(j-1)}$.
- **Arcs**: The arrows connecting places to transitions and transitions to places show the flow

of control or tokens. These arcs determine the direction of state changes within the system, illustrating how the robot moves from one state or place to another through the corresponding transitions. For example, an arc from p_{ij} to t_{ij} indicates the robot is ready to request permission to move to the next sector, while an arc from t_{ij} to $q_{i(j+1)}$ indicates the robot's movement to the next sector upon receiving permission.

- **Marking**: The marking of the Petri net represents the current state of the system, indicating the places that contain tokens. The initial marking M_0 is defined by the initial position of the robot A_i at the beginning of its path. The marking changes as the robot moves along its path, transitioning between different sectors and segments. The token in a place p_{ij} signifies the robot's presence in that sector, while the token in a place q_{ij} indicates the robot's entry to the segment p_{ij} with its disk overlapping both $p_{i(j-1)}$ and p_{ij} path segments.

The example paths for multiple robots are presented in Figures 3.2, and 3.3, and 3.4.

The Petri net model for the system with multiple robots based on equal length discretization is shown in Figure 3.8. The model is extended to include places and transitions for each robot A_i, where $i = 1, \ldots, m$. A similar model can be used for the optimal length discretization.

The Petri net model part for the details (see. Fig. 3.5) of the grid-based discretization is shown in Figure 3.9. The model is only shown for a part because the full model would be too large to fit on a single page. This is due to the fact that the grid-based discretization typically creates more segments than the equal length discretization or optimal length discretization. This can lead to more complex models and more complex control algorithms. In addition, the size of the cells in the grid structure can affect the efficiency of the control algorithm. If the cells are too large, the robot may not be able to move along the path as planned. If the cells are too small, the control algorithm may become too complex and require more computational resources.

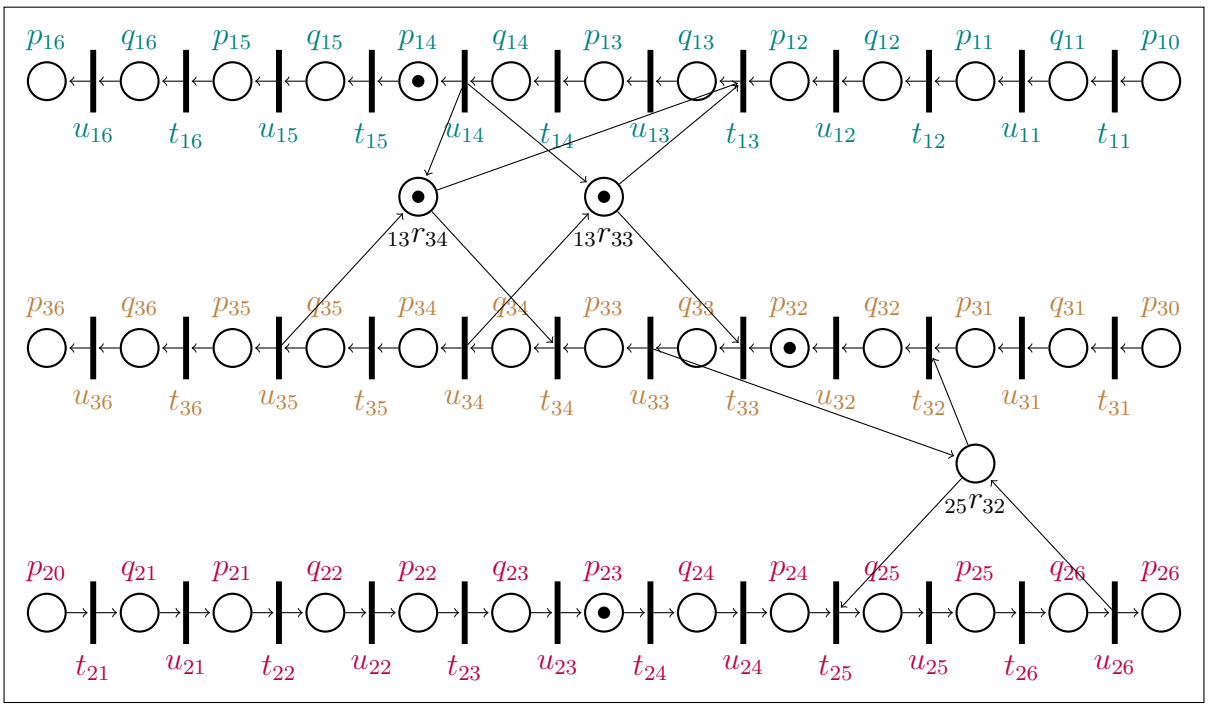


Figure 3.8: Petri net model for multiple robots based on equal length division

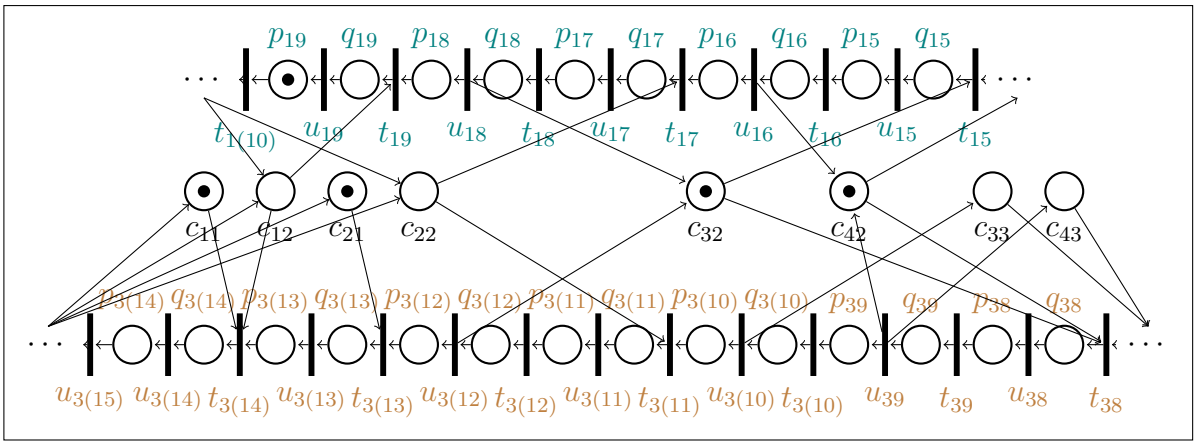


Figure 3.9: Petri net model for the detail of multiple robots based on grid division

3.3 Collision Avoidance

Collision avoidance is a critical aspect of robot motion control, ensuring that robots can navigate safely in their environment without colliding with obstacles or other robots. The system uses a simple collision avoidance algorithm that restricts the robot movement based on the conflict relation between the path segments determined through a direct path partition and resulting from the motion area partitioned into cells, respectively.

Definition 3 *For a set of n robot paths, respectively partitioned into n_i, $i = 1, \ldots, n$, sectors, two sectors are in conflict if they belong to two distinct paths and the minimal distance between these sectors is shorter than the diameter of the robot disk.*

Definition 4 *For a set of n robot paths, respectively partitioned into n_i, $i = 1, \ldots, n$, sectors, two sectors are in conflict if they belong to two distinct paths and they pass at least one common cell in the grid structure.*

To model such restrictions, it is necessary to add additional places that will represent the conflict segments of the paths.

The model in Figure 3.8 is extended to include resources ${}_{ij}r_{kl}$ modeled as places that represent the resources required to enter the conflict segments of the paths. These resources are necessary for robots to enter segments that collide with other paths. The proposed model also includes the situation when two consecutive segments of the same path are in conflict with the other path. This requires modeling the occupied segments of the paths as two separate resources ${}_{ij}r_{kl}$ and ${}_{ij}r_{k(l+1)}$ for the same path. The other robot can enter the segment p_{ij} only if the segments p_{kl} or $p_{k(l+1)}$ are not occupied [RJ21].

The model presented in Figure 3.9 is extended to include resources c_{xy} modeled as places that represent the cells in the grid structure that are required to be free to enter the conflict segments of the paths. The total number of resources c_{xy} is equal to the number of cells in the grid structure.

3.4 Deadlock Avoidance

Definition 5 *Deadlock is any situation in which no member of some group of entities can proceed because each waits for another member, including itself, to take action.*

There are two types of policies that deal with deadlocks in the systems: deadlock prevention and deadlock avoidance. The deadlock avoidance policy allows for more flexibility by detecting potential deadlock situations while the system is running and resolving them as they occur.

Definition 6 *In the considered P/T model of MMRS, marking M is ordered if there exists a permutation $z_1, z_2, \ldots, z_n$ of the set $N = \{1, 2, \ldots, n\}$ such that for each pair $(z_i, z_k) \in N$ the following is true. If $z_i \geq z_k$ then $M(p_{z_{k,j}}) = 0$ and $M(q_{z_{k,l}}) = 0$, where $p_{z_{k,l}}$ is any sector of robot A_{z_k} that is in conflict with any sector $p_{z_{i,j}}$ of robot A_{z_i} that lies between its currently occupied sector (defined by M) and the nearest non-conflict sector.*

Definition 7 *For each transition t enabled in any marking M, firing of t is safe if marking M' resulting from $M \xrightarrow{t} M'$ is ordered.*

To ensure that the robots can reach their final states without getting stuck it is necessary to reject the transitions $M \xrightarrow{t} M'$, such that M' is a deadlock state or unavoidably leads to a deadlock state. The identification of such transitions is a NP-complete problem. To avoid this complexity, the suboptimal deadlock avoidance policy is used. The policy is based on the modified version of the Banker's algorithm [Dij01], The modification involves changes in the maximum demand of each process. Instead of the constant maximal needs defined in the Banker's, the maximal needs depend on the marking and are the sum of the resources required for the current operation plus all the operations succeeding this one up to the nearest private resource. This modification reduces the restrictions on the resource requirements and allows more flexibility in the allocation of resources. The deadlock avoidance procedure takes the form of Algorithm 1, given below. The variables in the algorithm are defined as follows:

- Available is a vector of length (m) that represents the number of available resources of each type.
- Max_i is a matrix of size ($n \times m$) that represents the maximum demand of each process (n) for each resource type (m). The maximal needs are the sum of the resources required for the current operation plus all the operations succeeding this one up to the nearest private resource.
- Allocation_i is a matrix of size ($n \times m$) that represents the number of resources of each type (m) currently allocated to each process (n).
- Finish_i is an vector of size n that indicates whether each process (n) has completed its task.
- Work is a vector of length m that represents the number of resources available of each type after the completion of all the processes for which $\text{Finish}_i = true$.

3.5 Robot Model

The mobile robots are modeled as a differential drive robot, driven by two independently powered wheels on either side. The kinematic model of the robot describes its motion based on the velocities of the wheel, while the dynamic model considers the forces and torques acting on the robot.

The kinematic model of the mobile robots is defined by the following equations [SNS11] (see Fig. 3.10):

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \\ \dot{\theta} = \dfrac{v_r - v_l}{D} \end{cases} \tag{3.1}$$

Algorithm 1 Modified Banker's algorithm for deadlock avoidance

```
1:  Work ← Available
2:  Finish[i] ← false for i = 1, 2, ..., n
3:  while true do
4:      Safe ← false
5:      for i = 1 to n do
6:          if Finish[i] = false and Max[i] − Allocation[i] ≤ Work then
7:              Work ← Work + Allocation[i]
8:              Finish[i] ← true
9:              Safe ← true
10:         end if
11:     end for
12:     if Safe = false then
13:         break
14:     end if
15: end while
16: if Safe = true then
17:     return true
18: else
19:     return false
20: end if
```

where:

- x and y represent the robot's coordinates in the global frame,
- v is the linear velocity,
- θ is the heading angle,
- D is the distance between the wheels,
- v_r and v_l are the linear velocities of the right and left wheels, respectively.

The kinematic model describes the motion of the robot in terms of its linear velocities. The low-level robot motion control requires this to generate the appropriate wheel velocities to move the robot along the planned path.

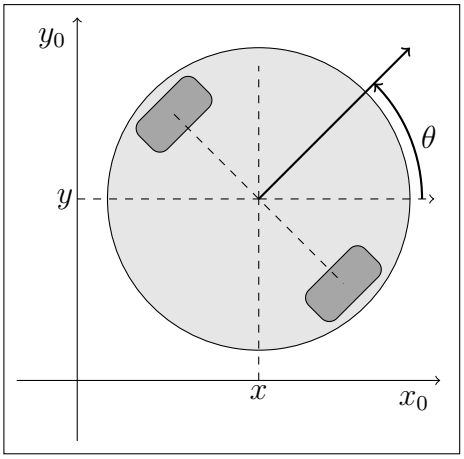


Figure 3.10: Differential drive robot model

Chapter 4

Implementation

The proposed system comprises multiple mobile robots managed by a central controller. The central controller handles inter-robot communication and mission assignments, while the robots execute the assigned missions, perform path planning, and follow designated paths. The robots utilized in this system are TurtleBot 2 models [Tur12, WF11]. The implementation leverages the Robot Operating System 2 (ROS2) framework [MFG+22], which is an open-source middleware that provides a structured communication layer above the robots' host operating systems. ROS2 is designed to be flexible and scalable, catering to a diverse range of robotic applications from low-level device control to high-level artificial intelligence [QCG+09]. Each robot functions as a node within the ROS network, communicating with the central controller via ROS messages. The central controller is also implemented as a separate node in the ROS network. The robots employ the ROS Navigation 2 Stack [MMWC20] for path planning and navigation.

4.1 System Architecture

The system architecture, illustrated in Figure 4.1, is based on a centralized control paradigm. The central controller (supervisor) manages the operations of all robots in the system, communicating with them over the ROS network. It receives mission assignments from the user and delegates these missions to individual robots. Constructed as a ROS2 package, the central controller comprises multiple nodes, each responsible for distinct tasks such as path discretization, collision avoidance, deadlock avoidance, and motion control. The central controller provides request-reply communication services to the robots, including path discretization and real-time decision-making for segment traversal.

Each robot navigator is also implemented as a ROS2 package. Robots act as nodes in the ROS network, using ROS messages to communicate with the central controller. They utilize the ROS Navigation 2 Stack for path planning and execution. The robots' low-level control systems handle movement and path-following, while high-level control systems manage communication with the central controller and navigation through segmented paths (e.g., entering a segment or waiting for it to be free).

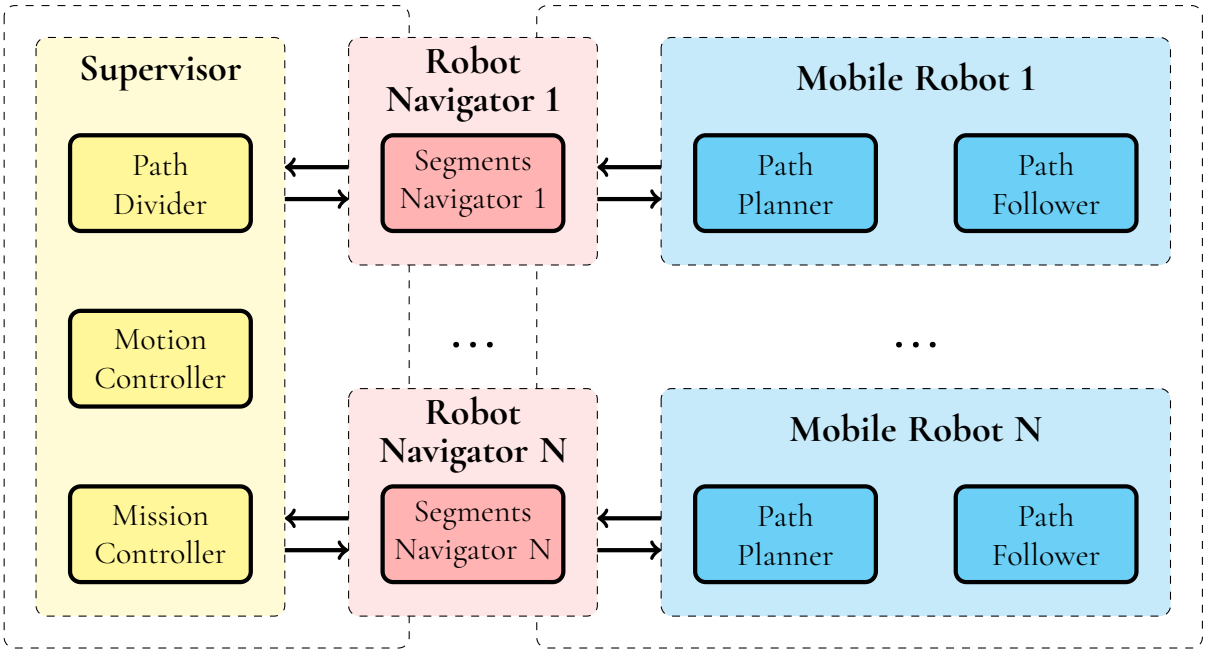


Figure 4.1: System architecture

4.2 Robot Movement

4.2.1 Path Planning

Path planning is a critical aspect of robot movement, allowing the robot to determine an optimal route from its starting position to its goal while avoiding obstacles. The system uses the ROS2 Navigation Stack, specifically the `nav2_planner` [MML+23] module, which provides several grid-based path planning algorithms. These algorithms take into account the robot's kinematic and dynamic constraints and the environment's layout. The path planner generates a series of waypoints that the robot must follow to reach its destination safely. The planner considers factors such as the robot's size, shape, and maximum velocity to ensure that the generated path is feasible for the robot to traverse.

To properly plan a path, the robot must have a map of its environment. Robots have the map of the environment that have been built using the SLAM algorithm beforehand. The map is represented as a grid, with each cell indicating whether it is occupied by an obstacle or free space. The path planner uses this map to determine the optimal path for the robot to follow.

The path planning module consists of two main components: the global planner and the local planner. The global planner generates a path for the robot to follow from its starting point to the destination. This path is computed using algorithms like A* or Dijkstra's algorithm, which consider the robot's environment as a grid map where each cell can be either free or occupied. The planner uses a costmap, which is a representation of the environment that includes information about obstacles, to ensure the path is collision-free. The local planner, or `nav2_controller` [MML+23], refines the global path and handles dynamic obstacles. It uses algorithms like Dynamic Window Approach (DWA) or Timed Elastic Band (TEB) to adjust the path in real-time. The local planner ensures the robot can navigate around unforeseen obstacles that were not present on the initial costmap used by the global planner.

4.2.2 Path Following

Path following [MSMG23] is the process by which the robot moves along the planned path. This involves converting the planned path into motor commands that drive the robot's wheels. The `nav2_controller` [MML+23] module translates the global path into a series of waypoints. The robot uses these waypoints to ensure it stays on the correct path. Controllers like Proportional-Integral-Derivative (PID) controllers or more advanced model predictive controllers (MPC) are used to generate appropriate wheel velocities. The robot constantly monitors its position using sensors such as LIDAR, IMU, and wheel encoders. The localization module uses solely AMCL (Adaptive Monte Carlo Localization) to estimate the robot's position and orientation in the map. This information is used to determine the robot's deviation from the path and make necessary corrections. Feedback mechanisms ensure the robot can correct its course in real-time, maintaining a smooth trajectory towards its goal.

4.3 Motion Control

The flow of the mission assignment is depicted in Figure 4.2. The motion control is illustrated in Figure 4.3. The central controller receives mission assignments from the user. If the assigned robot is registered with the central controller, the mission is accepted, otherwise rejected. The central controller instructs the robot to plan its path to the designated goal. Using the ROS Navigation 2 Stack, the robot plans its path and sends it back to the central controller. If not all robots have their missions assigned, the central controller stores the planned path for subsequent division until all of them send theirs paths to the central controller. This requirement is due to usage of the optimal segment discretization. The path can be optimally divided only if all the paths are known. Each path segment is uniquely identified and contains information about required resources, such as grid map cells or virtual resources when segments intersect. The supervisor then sends the divided paths to the robots. Each robot sequentially navigates through the path segments, requesting permission from the central controller to enter each one. The central controller checks if the segment is free and safe, especially to avoid potential deadlocks, and responds with either permission or rejection. If permission is denied, the robot periodically requests again. Upon receiving permission, the robot enters and traverses the segment, then requests the next one at the segment's end. This process continues until the robot reaches its goal. The low-level segment traversal communication is shown in Figure 4.4 and Figure 4.5.

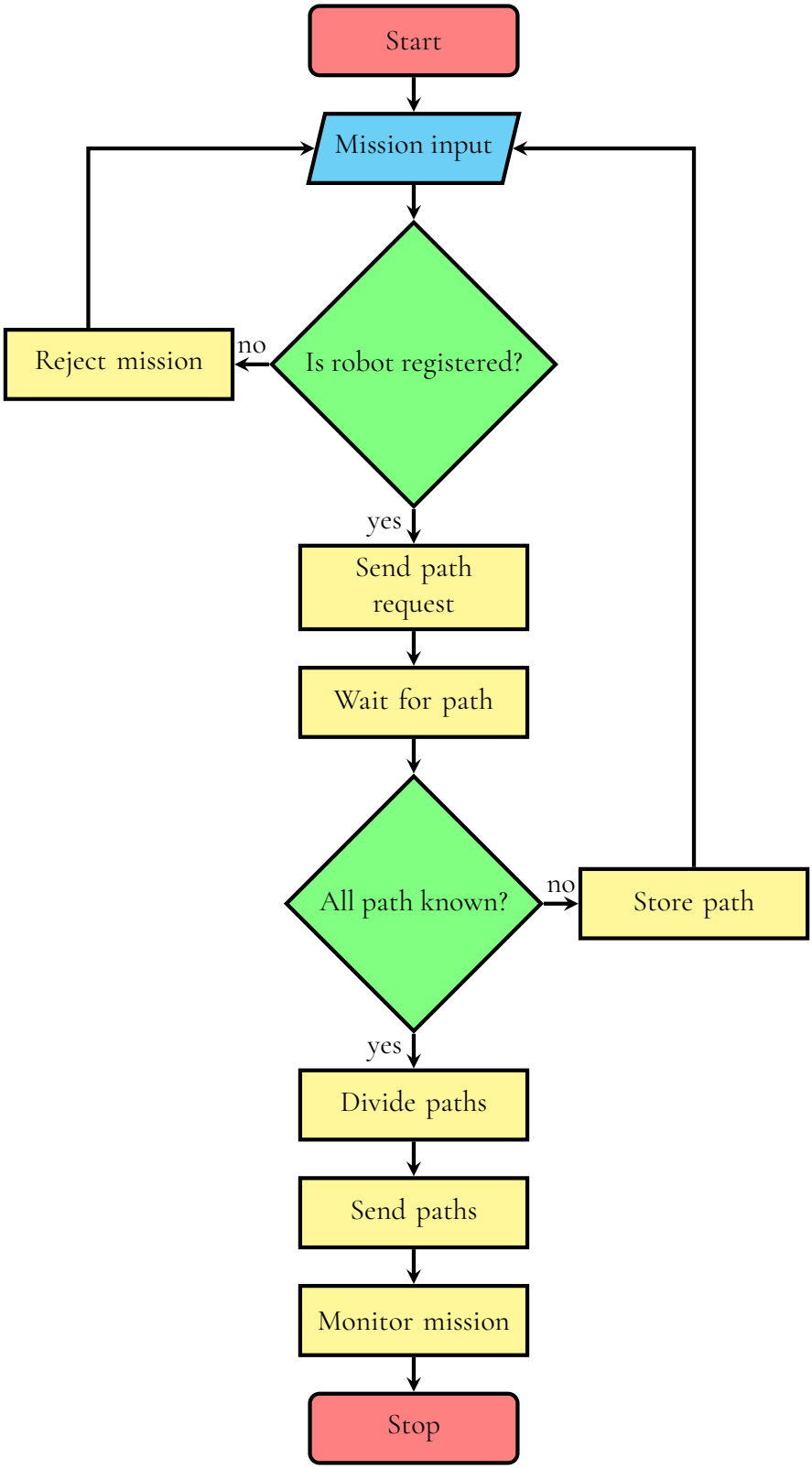


Figure 4.2: Mission assignment flow

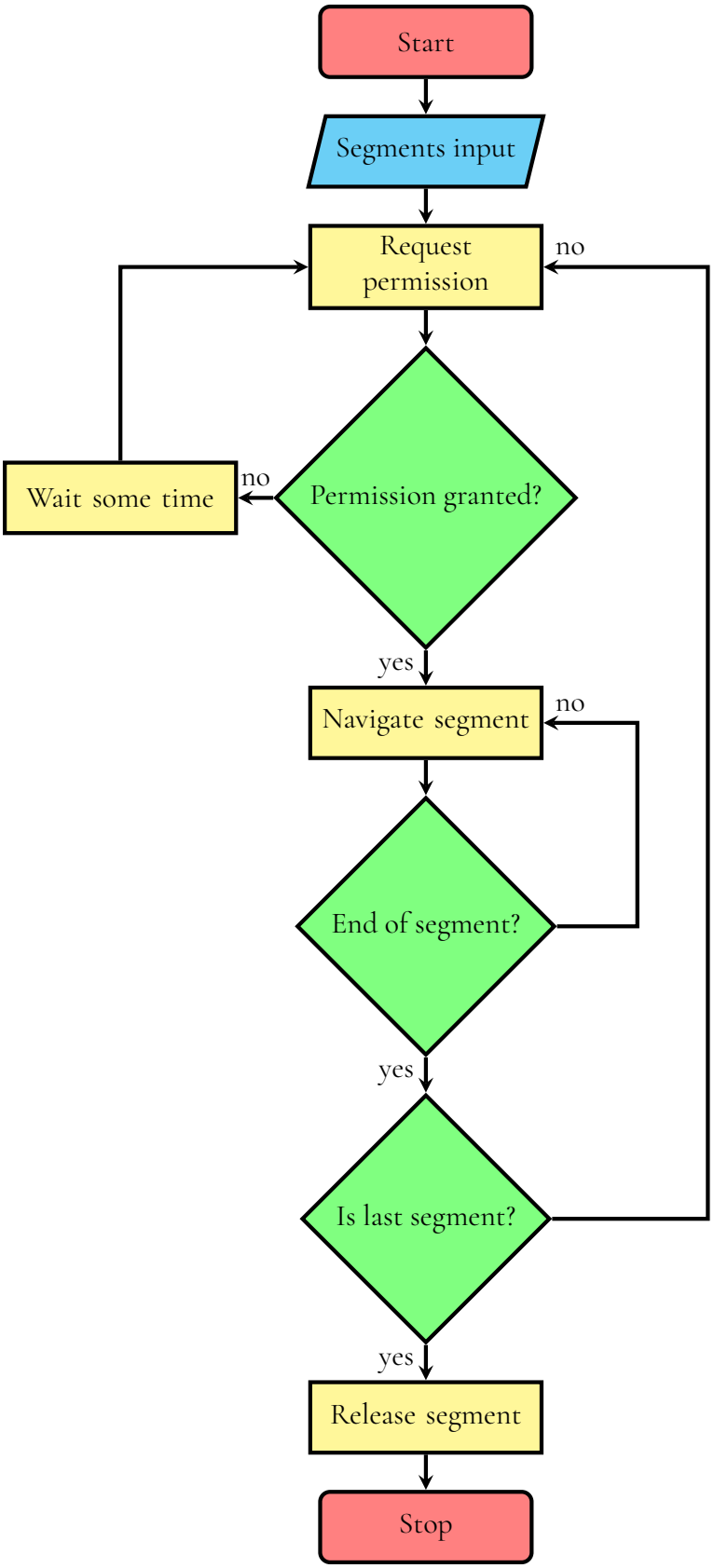


Figure 4.3: Motion control flow

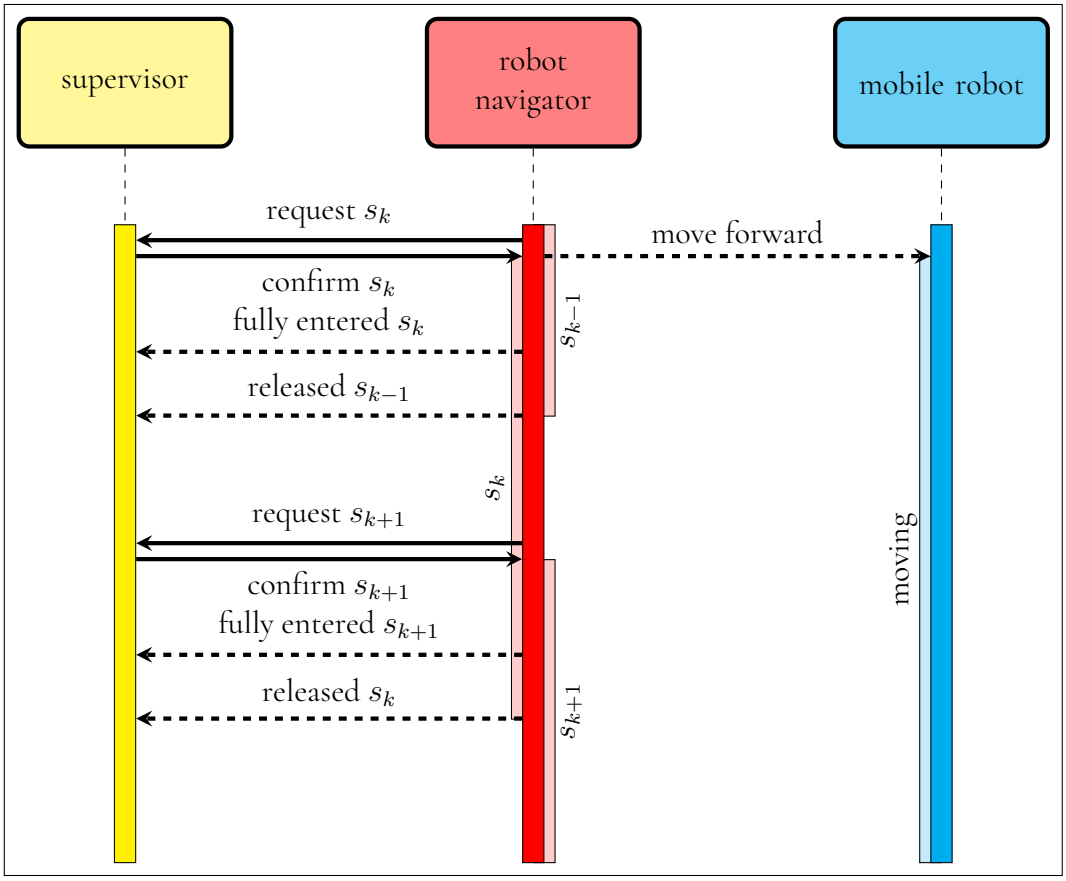


Figure 4.4: Segment traversal communication with no stop

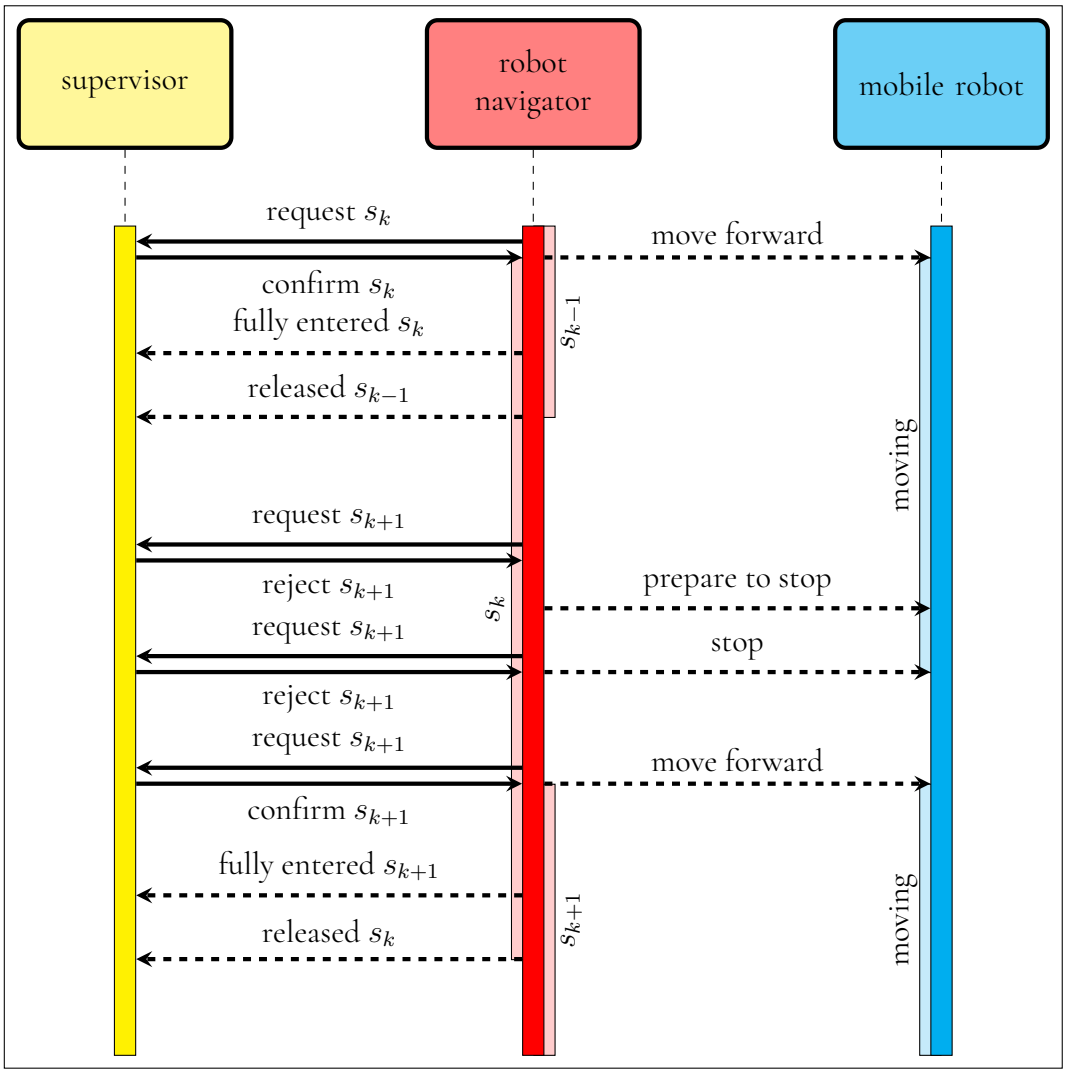


Figure 4.5: Segment traversal communication with stop

Chapter 5

Experiments and Results

This chapter delineates the findings from a series of experiments aimed at assessing the efficacy of a novel supervisory control strategy for multi-robot systems. The experiments were orchestrated in a static environment where the presence was limited to robots. The experimental setups involved configurations of two, three, and four robots, each receiving dynamic mission assignments. The results were meticulously evaluated across various configurations, focusing on collision scenarios and comparing the performance indicators for each experimental arrangement.

5.1 Detailed Description of the Experiments

5.1.1 Configuration of the Experiments

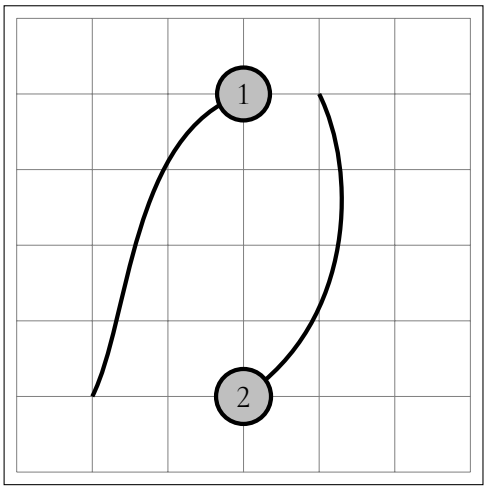
The experiments were designed methodically to determine the effectiveness of the supervisory control system within a controlled environment. In this static environment, obstacles were placed along the peripheries and robots were tasked with navigating through predefined waypoints. Three distinct configurations were employed:

1. **Two robots:** This initial setup was intended to evaluate the basic operational capabilities of the system in managing multiple robots simultaneously.
2. **Three robots:** Introducing an additional robot increased complexity, necessitating improved algorithms for collision avoidance and sophisticated path planning.
3. **Four robots:** This configuration tested the scalability and robustness of the system to coordinate multiple robotic units.

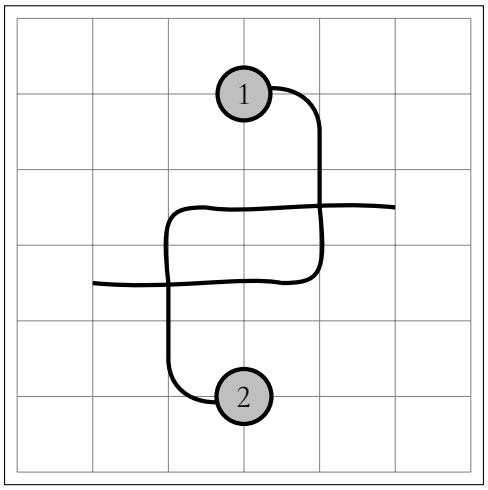
The segmentation length for equal path division was established at 1.2 meters, while a grid size of 0.8 meters was utilized for the grid division method.

5.1.2 Robot Paths

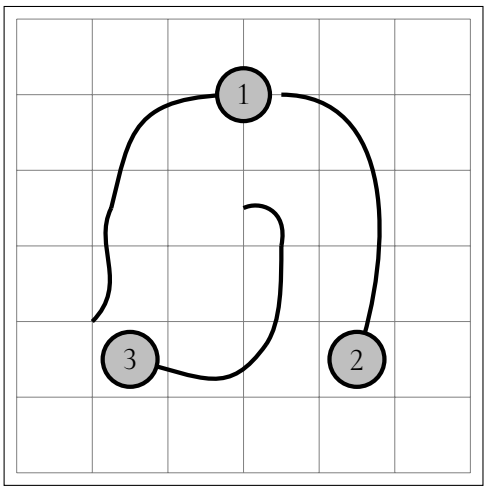
The trajectories executed by the robots in different configurations are shown in Figure 5.1. Each subfigure visually represents the paths for varying numbers of robots, illustrating scenarios both with and without collisions.



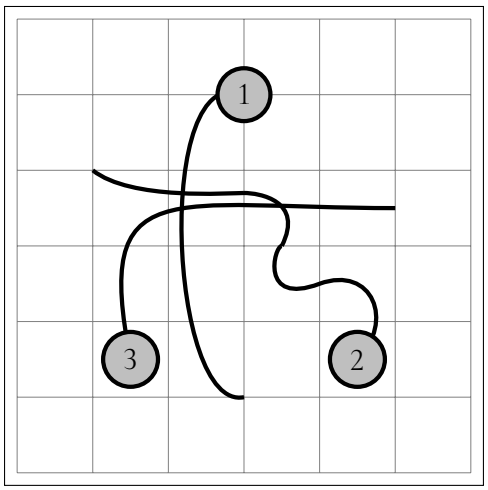
(a) Paths of 2 robots with no collisions



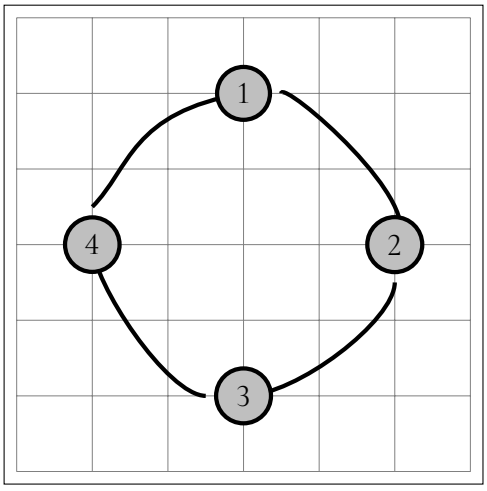
(b) Paths of 2 robots with collisions



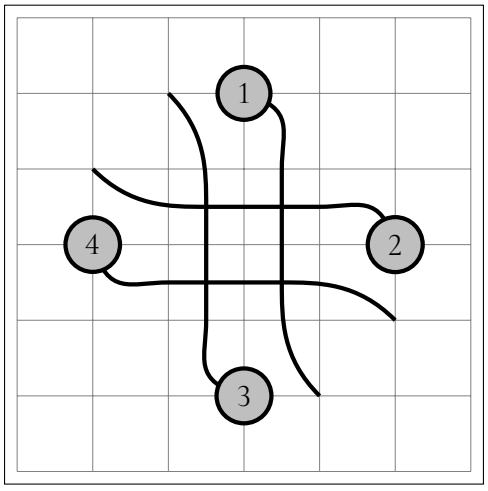
(c) Paths of 3 robots with no collisions



(d) Paths of 3 robots with collisions



(e) Paths of 4 robots with no collisions



(f) Paths of 4 robots with collisions

Figure 5.1: Paths of robots used in experiments

5.2 Presentation of Results

The results of the experiments are systematically presented in tabular formats, detailing performance metrics such as mission duration, planning time, segmentation time, the total number of segments, and the aggregate length of paths. Each table contrasts the results of different path division strategies: equal, optimal, and grid.

5.2.1 Outcomes for Two Robots in Non-collision Scenarios

The results, as shown in Table 5.1, indicate that the optimal division method consistently yielded the shortest mission durations compared to the equal and grid methods. This suggests that the optimal method can effectively minimize the complexity of movement, thereby facilitating faster completion of missions. The absence of collisions allowed the optimal division to uniformly divide the path into one segment, ensuring uninterrupted movement for the robots. In contrast, the grid method, despite its structured approach, resulted in the longest mission durations due to an increased number of segments. In particular, in scenarios involving short paths without collisions, all methods demonstrated similar segmentation times.

Table 5.1: Results of the experiments with 2 robots and no collisions

division type	robot	mission time [s]	planning time [s]	division time [s]	segments [#]	length [m]
equal	1	22.09	0.04	0.02	4	4.25
	2	19.07	0.04	0.02	4	4.28
optimal	1	22.03	0.05	0.03	1	4.22
	2	20.13	0.03	0.03	1	4.25
grid	1	26.27	0.04	0.03	22	4.23
	2	22.07	0.04	0.03	20	4.28

5.2.2 Outcomes for Three Robots in Non-collision Scenarios

In scenarios involving three robots, the optimal division method once again proved superior by demonstrating the lowest mission times, underscoring its efficacy in dynamic mission allocation and execution. The equal division method exhibited moderate performance, while the grid method, due to its extensive segmentation, incurred longer mission times. The results, detailed in Table 5.2, show an increase in the division times for the optimal method, attributed to the increased complexity introduced by the additional robot. The other methods maintained division times comparable to those of previous setups.

5.2.3 Outcomes for Four Robots in Non-collision Scenarios

Similar trends were observed with four robots, where the optimal division method consistently led to the most efficient outcomes in terms of the duration of the mission. The results, encapsulated in Table 5.3, reveal that while the equal division method provided moderate results, the grid method was the least efficient, requiring the highest mission times due to a larger number of segments.

Table 5.2: Results of the experiments with 3 robots and no collisions

division type	robot	mission time [s]	planning time [s]	division time [s]	segments [#]	length [m]
equal	1	27.07	0.06	0.03	4	4.28
	2	24.87	0.09	0.03	4	4.40
	3	37.16	0.10	0.03	5	5.55
optimal	1	25.35	0.06	0.08	1	4.22
	2	21.62	0.06	0.08	1	4.37
	3	29.77	0.05	0.08	1	5.51
grid	1	37.54	0.06	0.04	22	4.25
	2	35.80	0.04	0.04	23	4.40
	3	48.26	0.04	0.04	27	5.51

The division times for the optimal method remained relatively stable, whereas those for the other methods were comparable to those in simpler scenarios.

Table 5.3: Results of the experiments with 4 robots and no collisions

division type	robot	mission time [s]	planning time [s]	division time [s]	segments [#]	length [m]
equal	1	31.32	0.12	0.02	3	3.26
	2	24.70	0.06	0.02	3	3.24
	3	26.23	0.07	0.02	3	3.32
	4	37.37	0.05	0.02	3	3.25
optimal	1	22.33	0.04	0.05	1	2.82
	2	16.78	0.06	0.05	1	2.82
	3	18.63	0.05	0.05	1	2.82
	4	21.63	0.04	0.05	1	2.85
grid	1	26.16	0.04	0.03	16	3.27
	2	28.47	0.05	0.03	15	3.25
	3	37.64	0.05	0.03	22	3.33
	4	27.16	0.06	0.03	18	3.25

5.2.4 Outcomes for Two Robots in Collision Scenarios

In collision scenarios involving two robots, the results from Table 5.4 demonstrate a marked impact of collisions on mission times. Notably, the optimal division method again exhibited the

shortest mission times, highlighting its capability to effectively manage collisions. Although the equal division method aligned the number of segments with the optimal method, it resulted in longer mission durations due to the higher number of segments that collide with each other, thereby necessitating additional waiting times for collision avoidance. In contrast, the grid method, despite its structured segmentation, incurred the longest mission times owing to the larger number of segments. The introduction of collisions notably increased the division times for the optimal method, while the division times for the other methods remained consistent with non-collision scenarios.

Table 5.4: Results of the experiments with 2 robots and collisions

division type	robot	mission time [s]	planning time [s]	division time [s]	segments [#]	length [m]
equal	1	54.89	0.04	0.03	5	5.43
	2	34.08	0.06	0.03	5	5.33
optimal	1	38.73	0.04	0.06	5	5.38
	2	34.48	0.06	0.06	5	5.36
grid	1	57.97	0.04	0.04	30	5.43
	2	49.80	0.06	0.04	25	5.33

5.2.5 Outcomes for Three Robots in Collision Scenarios

The dynamics of three robots experiencing collisions present varied results, as detailed in Table 5.5. Contrary to non-collision scenarios, the optimal method did not lead but showed moderate mission times, whereas the equal division method exhibited the best performance due to effective path placements. The optimal method, despite having the fewest segments, faced challenges as robots had to wait for clearance due to longer collision segments compared to other scenarios, thus affecting mission durations. The grid method, consistently producing the longest mission times and highest segment count, also demonstrated inefficiencies in the effective handling of collisions. Division times in the optimal method were significantly higher, reflecting the complexity added by collisions, while the other methods showed division times similar to their non-collision counterparts.

5.2.6 Outcomes for Four Robots in Collision Scenarios

The scenario involving four robots with collisions, represented in Table 5.6, is indicative of the complexity and challenges posed by multiple robots operating in confined spaces with potential conflicts. The optimal division method, though not the most efficient in segment number, resulted in better performance in terms of mission time compared to equal division. This suggests that while the optimal method managed the paths more effectively during collisions, the equal division method did not adequately account for proximity and potential path interferences, resulting in more frequent collisions and, consequently, longer mission times. The grid method once again resulted in the most prolonged mission times and highest segment counts, underscoring its lack of efficiency in collision-rich environments. The division times for the optimal method peaked due

Table 5.5: Results of the experiments with 3 robots and collisions

division type	robot	mission time [s]	planning time [s]	division time [s]	segments [#]	length [m]
equal	1	26.17	0.08	0.03	4	4.54
	2	51.87	0.05	0.03	5	6.01
	3	36.35	0.07	0.03	6	6.26
optimal	1	67.06	0.06	0.11	3	4.55
	2	49.63	0.10	0.11	3	5.98
	3	40.40	0.09	0.11	3	6.24
grid	1	45.69	0.04	0.04	21	4.58
	2	72.22	0.06	0.04	36	6.04
	1	49.70	0.05	0.04	37	6.26

to the intricate calculations required to manage the increased robot count and collision possibilities, whereas the other methods maintained consistent division times as seen in scenarios without collisions.

Table 5.6: Results of the experiments with 4 robots and collisions

division type	robot	mission time [s]	planning time [s]	division time [s]	segments [#]	length [m]
equal	1	65.24	0.04	0.03	4	4.56
	2	32.06	0.07	0.03	4	4.67
	3	31.85	0.09	0.03	4	4.60
	4	31.72	0.06	0.03	4	4.59
optimal	1	36.73	0.04	0.12	5	4.60
	2	28.32	0.05	0.12	5	4.65
	3	28.69	0.07	0.12	5	4.57
	4	32.86	0.05	0.12	5	4.57
grid	1	49.05	0.06	0.04	23	4.62
	2	67.31	0.26	0.04	21	4.67
	3	58.95	0.04	0.04	24	4.61
	4	42.80	0.04	0.04	25	4.60

5.3 Analysis and Discussion

The experimental investigations elucidated the distinct capabilities and limitations inherent to each path division strategy:

- **Equal Division Method:** Characterized by its simplicity, this approach performs competently in scenarios with minimal complexity and no collisions. It encounters difficulties in more complex environments that include multiple robots and potential collisions, resulting in extended mission durations. The calculation times remain the lowest and consistent across all tests due to the straightforward nature of the computations.

- **Optimal Division Method:** This strategy is particularly effective at minimizing mission times in environments devoid of collisions through adept path segmentation. Despite a decrease in performance when collisions occur, it still maintains superiority over other methods. The computation times are elevated, reflecting the intricacies of the segmentation process. As the scenarios become more complex with more robots, the time required for path division increases. In particular, this method consistently uses the fewest segments among the tested approaches.

- **Grid Division Method:** This method generally leads to the longest mission durations because of its extensive segmentation, which also introduces added complexity to the overall model. Although the division times are comparable to those of the equal division method, the number of segments required far exceeds those of the other strategies.

In conclusion, the choice of path division strategy should depend on the specific demands of the mission and the operational dynamics of the environment. The optimal division method is preferable in complex scenarios where efficiency is paramount, whereas the equal division method suffices for less complicated situations. The findings from these experiments are pivotal in guiding the selection of an appropriate strategy, ensuring both efficiency and adaptability in the supervisory control of multi-robot systems. All of the methods provide a solution in which robots do not collide with each other. The addition of the deadlock avoidance mechanism ensures that robots do not get stuck in a deadlock situation. All robots in all configurations were able to reach their destinations without any deadlock situations.

Chapter 6

Summary

This study addresses the development of models and algorithms for the supervisory control of multiple mobile robot systems (MMRS), with an emphasis on ensuring safe navigation and effective coordination among robots. The primary objective is to improve the operational efficiency and reliability of MMRS in static environments by employing sophisticated control strategies and algorithms.

The research introduces a hierarchical control framework comprising three levels: high-level supervisory control, mid-level stage traverse control, and low-level robot motion control. This framework integrates discrete event systems (DES) and continuous-time systems (CTS), enabling a comprehensive approach to managing the complex tasks and movements of multiple robots. The high-level control focuses on mission planning and task allocation, while mid-level control oversees the transitions between discrete path segments and continuous motion. Low-level control is responsible for the precise execution of robot movements, using continuous feedback to ensure accurate path follow-up and obstacle avoidance.

Path discretization plays a critical role in the coordination of robot movements, with three primary methods explored: equal length, optimal length, and grid-based discretization. Each method has been evaluated for its effectiveness in minimizing movement time for whole mission. The study highlights the advantages of optimal length discretization for its ability to handle complex environments by segmenting paths by maximizing the length of non-collision segments and minimizing the number of transitions between segments.

Collision and deadlock avoidance policies are addressed through advanced algorithms and control mechanisms. The research employs Petri net models to represent the discrete states and events associated with robot movements, ensuring that robots operate without collisions and deadlocks. The modified Banker's algorithm is utilized for online deadlock avoidance, providing a dynamic method to detect and resolve potential situations that unavoidably lead to deadlocks as they occur. This approach enhances the flexibility and robustness of the MMRS, allowing for safe and efficient navigation even in unpredictable environments.

The experimental implementation of the proposed control framework was conducted using TurtleBot 2 robots within the Robot Operating System 2 (ROS2) framework. The system architecture integrates a central controller that manages robot communication, mission assignments, and real-time decision-making. Each robot operates as a node within the ROS network, employing the ROS Navigation 2 Stack for path planning and execution. The experimental results demonstrate the effectiveness of the proposed control strategies in various scenarios, including collision and deadlock avoidance, highlighting the potential of the system developed for practical applications.

In conclusion, this study contributes to the field of robotics by providing innovative solutions

for the supervisory control of MMRS.The proposed hierarchical and hybrid control framework, along with advanced algorithms for path discretization, collision avoidance, and deadlock prevention, offers a robust approach to managing multi-robot systems in complex environments. Future research could focus on further optimizing these algorithms and exploring their applications in different robotic settings, enhancing the capabilities and reliability of multi-robot systems for a wide range of tasks and missions.

Availability of the project

The code and documentation for the project can be found at https://gitlab.com/jakub.kozlowicz/mmrs. The repository has been published under the MIT License, allowing for the free use and modification of the software for research and educational purposes.

Bibliography

[ACF+98] Rachid Alami, Raja Chatila, Sylvain Fleury, Malik Ghallab, and François Ingrand. An architecture for autonomy. *The International Journal of Robotics Research*, 17(4):315–337, 1998.

[Ark98] Ronald C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, MA, 1998.

[Bad15] Prashant Badoni. Robotics Controller: A Literature Survey. *International Journal of Engineering Research & Technology (IJERT)*, 4(10), October 2015.

[BQ12] L. Bruzzone and G. Quaglia. Review article: locomotion systems for ground mobile robots in unstructured environments. *Mechanical Sciences*, 3(2):49–62, 2012.

[BR07] James G. Bellingham and Kanna Rajan. Robotics in remote and hostile environments. *Science*, 318(5853):1098–1102, 2007.

[Bro91] Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47(1–3):139–159, 1991.

[BSP23] Matteo Bettini, Ajay Shankar, and Amanda Prorok. Heterogeneous multi-robot reinforcement learning, 2023.

[BV00] Michael Bowling and Manuela Veloso. Motion control in dynamic multi-robot environments. In Manuela Veloso, Enrico Pagello, and Hiroaki Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, pages 222–230, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[C+92] R Craig Coulter et al. *Implementation of the pure pursuit path tracking algorithm*. Carnegie Mellon University, The Robotics Institute, 1992.

[CL10] Christos G Cassandras and Stéphane Lafortune. *Introduction to Discrete Event Systems*, page 800. 01 2010.

[Cor17] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer, Berlin, Heidelberg, 2017.

[Dij01] Edsger W Dijkstra. Solution of a problem in concurrent programming control. In *Pioneers and Their Contributions to Software Engineering: sd&m Conference on Software Pioneers, Bonn, June 28/29, 2001, Original Historic Contributions*, pages 289–294. Springer, 2001.

[DJ10] Gregory Dudek and Michael Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, Cambridge, 2010.

[DMG15] Rajesh Doriya, Siddharth Mishra, and Swati Gupta. A brief survey and analysis of multi-robot communication and coordination. In *International Conference on Computing, Communication & Automation*, pages 1014–1021, 2015.

[FHSS20] Zhi Feng, Guoqiang Hu, Yajuan Sun, and Jeffrey Soon. An overview of collaborative robotic manipulation in multi-robot systems. *Annual Reviews in Control*, 49:113–127, 2020.

[FIN04] A. Farinelli, L. Iocchi, and D. Nardi. Multirobot systems: a classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(5):2015–2028, 2004.

[FN87] Toshio Fukuda and S. Nakagawa. Dynamically reconfigurable robotic system. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1581–1586, 1987.

[GM04] Brian P. Gerkey and Maja J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

[GM12] Avinash Gautam and Sudeept Mohan. A review of research in multi-robot systems. In *2012 IEEE 7th International Conference on Industrial and Information Systems (ICIIS)*, pages 1–5, 2012.

[HAPG21] Miguel Hernando, Mercedes Alonso, Carlos Prados, and Ernesto Gambao. Behavior-based control architecture for legged-and-climber robots. *Applied Sciences*, 11(20), 2021.

[IPS+19] Marina Indri, Corrado Possieri, Fiorella Sibona, Pangcheng David Cen Cheng, and Vinh Duong Hoang. Supervised global path planning for mobile robots with obstacle avoidance. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 601–608, 2019.

[JW98] Nicholas R. Jennings and Michael Wooldridge. *Agent Technology: Foundations, Applications, and Markets*. Springer, Berlin, Heidelberg, 1998.

[KFS05] Nidhi Kalra, Dave Ferguson, and Anthony Stentz. Hoplite: A market-based framework for planned tight coordination in multirobot teams. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1170–1176, 2005.

[KHE15] Alaa Khamis, Ahmed Hussein, and Ahmed Elmogy. *Multi-robot Task Allocation: A Review of the State-of-the-Art*, pages 31–51. Springer International Publishing, Cham, 2015.

[KKB10] Dov Katz, Jacqueline Kenney, and Oliver Brock. How can robots succeed in unstructured environments? *Citeseer*, 12 2010.

[LaV06] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, 2006.

[MFG+22] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.

[MG10] John Moore and Stephen Gloss. Supervisory control of multiple robots. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 40(1):35–44, 2010.

[MML+23] Steve Macenski, Tom Moore, David V. Lu, Alexey Merzlyakov, and Michael Ferguson. From the desks of ros maintainers: A survey of modern & capable mobile

robotics algorithms in the robot operating system 2. *Robotics and Autonomous Systems*, 168:104493, 2023.

[MMWC20] Steve Macenski, Francisco Martín, Ruffin White, and Jonatan Ginés Clavero. The Marathon 2: A Navigation System. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2718–2725, 2020.

[MSMG23] Steve Macenski, Shrijit Singh, Francisco Martín, and Jonatan Ginés. Regulated pure pursuit for robot path tracking. *Autonomous Robots*, 47(6):685–694, Aug 2023.

[Mur04] Robin R. Murphy. Trial by fire [rescue robots]. *IEEE Robotics & Automation Magazine*, 11(3):50–61, 2004.

[Par08] Lynne E. Parker. Distributed intelligence: Overview of the field and its application in multi-robot systems. *Journal of Physical Agents*, 2(1):5–14, 2008.

[PR13] David Portugal and Rui P. Rocha. Multi-robot patrolling algorithms: examining performance and scalability. *Advanced Robotics*, 27(5):325–336, 2013.

[QB03] S.Joe Qin and Thomas A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, 2003.

[QCG+09] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[RAT18] Yara Rizk, Mariette Awad, and Edward W. Tunstel. Decision making in multiagent systems: A survey. *IEEE Transactions on Cognitive and Developmental Systems*, 10(3):514–529, Sep. 2018.

[RAT19] Yara Rizk, Mariette Awad, and Edward W. Tunstel. Cooperative heterogeneous multi-robot systems: A survey. *ACM Comput. Surv.*, 52(2), apr 2019.

[RJ21] Elzbieta Roszkowska and Janusz Jakubiak. Control synthesis for multiple mobile robot systems. *Transactions of the Institute of Measurement and Control*, 0(0):01423312211047061, 2021.

[RMCJ23] Elzbieta Roszkowska, Piotr Makowski-Czerski, and Lukasz Janiec. Multi-level control for multiple mobile robot systems. *Discrete Event Dynamic Systems*, 33(4):425–453, Dec 2023.

[SK16] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer, Berlin, Heidelberg, 2016.

[SLGR03] J.D. Sweeney, Huan Li, R.A. Grupen, and K. Ramamritham. Scalability and schedulability in large, coordinated, distributed robot systems. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 3, pages 4074–4079 vol.3, 2003.

[SNS11] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[SV00] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, Jun 2000.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.

[TMD+06] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692, 2006.

[TME14] Philip Twu, Yasamin Mostofi, and Magnus Egerstedt. A measure of heterogeneity in multi-agent systems. In *2014 American Control Conference*, pages 3972–3977, 2014.

[Tur12] TurtleBot. Turtlebot 2. https://www.turtlebot.com/turtlebot2/, 2012. [accessed 2024-06-01].

[Utk92] Vadim I. Utkin. Sliding modes in control and optimization. In *Communications and Control Engineering Series*, 1992.

[WF11] Melonee Wise and Tully Foote. Specification for turtlebot compatible platforms. https://www.ros.org/reps/rep-0119, Dec 2011. [accessed 2024-06-01].

[WRC23] Liyana Wijayathunga, Alexander Rassau, and Douglas Chai. Challenges and solutions for autonomous ground robot scene understanding and navigation in unstructured outdoor environments: A review. *Applied Sciences*, 13(17), 2023.

[YSSA19] Mark Yim, Wei-Min Shen, David Saldaña, and Masoud Asadpour. Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 26(4):22–30, 2019.

[ZHL+18] Yuan Zhou, Hesuan Hu, Yang Liu, Shang-Wei Lin, and Zuohua Ding. A distributed approach to robust control of multi-robot systems. *Automatica*, 98:1–13, 2018.

[ZHLD17] Yuan Zhou, Hesuan Hu, Yang Liu, and Zuohua Ding. Collision and deadlock avoidance in multirobot systems: A distributed approach. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(7):1712–1726, 2017.