A data grid strategy for non-prehensile object transport by a multi-robot system

by

© **Priyank Narvekar**

A thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of Master of Science.

Department of Computer Science

Memorial University

March 2024

St. John's, Newfoundland and Labrador, Canada

Abstract

The field of cooperative object transport within swarm and multi-robot systems (MRS) is an interesting area of research because exploring the dynamics of how multiple robots collaborate to transport objects presents fascinating opportunities to advance the capabilities of robotic teams.

In this thesis we propose a control framework for non-prehensile object transport using a multi-robot system. While an object can be unmanageable for a single robot to push and transport, we demonstrate via simulations that a team of cooperative robots can be used to transport such an object. The proposed control strategy is divided into two phases: caging and cooperative transport. In the first phase, the robots start from arbitrary positions and then approach the object to be transported, forming a cage around it. The second phase consists of cooperatively transporting the object ensuring that it remains caged during transport.

In the proposed strategy, the robots take a decentralized approach, wherein each robot operates autonomously while maintaining indirect communication with other robots. This is achieved by utilizing distributed data structures, such as distributed locks, sets, and maps, offered by the concept of an in-memory data grid (IMDG). By leveraging these distributed data structures, the robots can effectively share their respective states with one another.

The key advantage of employing distributed data structures within our strategy is that it eliminates the need to develop a new application-specific protocol for inter-robot communication. Instead, we can take advantage of the existing functionality provided by the data grid, which offers a convenient mechanism for exchanging information between robots.

To our knowledge, the utilization of an IMDG in the domain of MRS is a novel

concept. This thesis introduces a proposed design for a coordinated motion control strategy specifically aimed at object transport. The strategy leverages the capabilities of an IMDG, and presents it as a promising framework to simplify the communication process among the robots involved, leading to improved coordination and optimized object transport operations.

We showcase the results of our research using a realistic simulator that effectively illustrates the viability of our method. This demonstration helps validate the effectiveness and potential of our methodology in various environments.

Acknowledgements

I want to express my gratitude to my supervisor, Prof. Andrew Vardy, for giving me the opportunity to embark on this degree program through his acceptance into the esteemed Bio-inspired Robotics Lab (BOTS). His guidance, indispensable advice, and unwavering support have been instrumental in shaping my academic journey at graduate school. I am also grateful to the fellow students in the lab for their shared learning experiences.

Additionally, I would like to extend my heartfelt appreciation to my parents and family for their encouragement. Their constant support has been a pillar of strength throughout this endeavor.

Table of contents

List of figures

Chapter 1

Introduction

There has been a growing focus on cooperative MRS in recent years because they can be made up of multiple simple, economical, and repurposable robots that are easier to build and program than a single complex, application-specific robot [18, 5, 65, 28, 27, 55]. In parallel, there has also been an increasing interest in the use of distributed computing systems due to easier scaling, increased fault-tolerance performance and hence increase cost-effectiveness and flexibility [11, 32, 54]. Cooperative object transport is useful for many tasks such as gathering dispersed objects or transporting objects to a secondary system that can optimally store them or assemble them [61, 4]. In this research we make use of a particular type of distributed system— in-memory data grid (IMDG) —for the purpose of co-operative object transport. In our approach, the robots do not grasp the object to be transported (prehensile manipulation), but rather push the object (nonprehensile manipulation). This reduces the complexity of the robots required to implement this solution.

MRS are designed to work together as a team, hence effective and efficient communication is essential for the robots to share information, coordinate their actions,

and perform tasks more efficiently. IMDG is a distributed computing technology that provides fast access and processing times for shared data [26]. Rather than developing a new communication protocol, our use of IMDG allows us to focus on solving the problem at hand while counting on the data grid to manage how state information is shared and how collective decisions are made.

An IMDG system works by running a dedicated process on every node or robot in the group, allowing collaborative application access to the necessary data. Each robot in the cluster maintains its own unique view of the data which is held in memory, but these views are made available to all other members of the group. The process is responsible for monitoring all the data on each robot, allowing it to be shared with any other robots. An IMDG allows the application to abstract away the complexity of coordination and orchestration required to retrieve and update data across the network, simplifying application development [26, 15, 68, 48]. Several IMDG platforms are available, including open source projects. These include Hazelcast, Apache Ignite, Infinispan, Redhat Jboss Data Grid and others [20]. For the purpose of this thesis, Hazelcast was chosen because of its popularity and simplicity.

The objective of this thesis is to design a coordinated motion control strategy for object transport using nonprehensile robots that leverage IMDG for inter-robot communication. For the purpose of validation, we demonstrate the use of this strategy through realistic simulations in which robots move a cube through an environment, even in the presence of walls that obstruct their path.

1.1 Problem description

The fundamental objective of the cooperative transport problem of this research revolves around the collaborative effort of a team of robots to move a polygonal object to a predetermined destination. This object is larger than the individual robots themselves, necessitating their coordinated push to move it from its initial position towards the specified goal location. Once the robots have received the information about the object to be transported to a goal, their task is to approach the object and work together to propel the object towards the intended destination. This has been depicted in the Figure 2.1 in which we have a supervisory system, which is provided with the desired state and perceives the current state. The system identifies the team of bots b1, b2, b3 and b4 for the transportation task of object o1. Once the bots receive the control message from the supervisor, they form a cluster. b2 then assumes the interim leader role and the robots approach the object, enclose it and then transport it to the target.

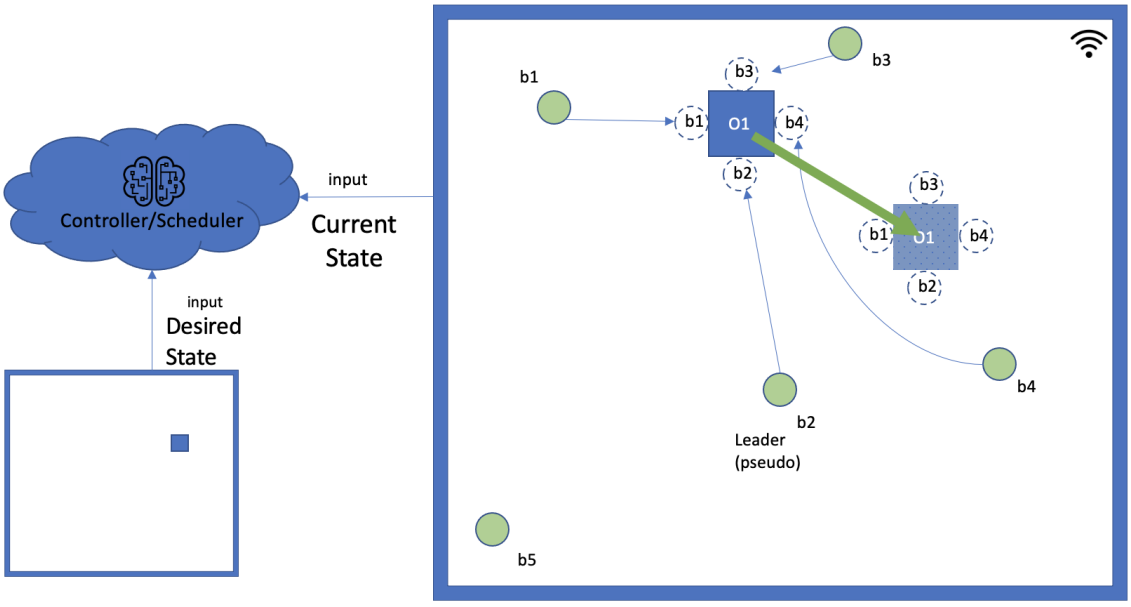


Figure 1.1: Diagram depicting the problem and proposed control framework

1.2 Scope and Objectives

The objectives of this research and thesis is to get robots intentionally cooperate with each other to transport an object from its original position to a desired position. While doing so, robots have to establish and use a local IMDG cluster as a means of communication. To achieve this goal, we have made the following assumptions about the robot and the environment.

- Robots have omni directional navigational capabilities.
- Robots use explicit and direct interaction.
- Robots can avoid static obstacles but not dynamic obstacles.
- Robots are non-prehensile in nature, i.e. they can only push the object to manipulate it, not grasp or seize them.

These assumptions are made to streamline the design and operation of robotic systems within a time frame feasible for a master's program. Omni-directional navigational capabilities simplify path planning and movement in complex spaces. Focusing on avoiding stationary objects allows us to first verify the viability of the idea of using IMDG as a means of communication, without getting involved in the complexity of sensor data sharing in order to avoid dynamic obstacles. Lastly, assuming non-prehensile manipulation capabilities reduces complexity in robotic manipulation tasks, particularly in scenarios where object movement is prioritized over intricate manipulation. Overall, these assumptions aim to strike a balance between simplifying system design while still achieving the aforementioned objectives.

1.3 Contribution

In this research, we introduced the following innovative and novel concept :

- Usage of imdg for inter-robot communication and coordination in a multi-robot system

First, the utilization of an in-memory data grid by existing software implementations as a communication provider for inter-robot communication has been proposed. This approach leverages the efficiency and speed of an existing in-memory data grid technology to simplify communication between robots, enabling seamless coordination and collaboration. This thesis emphasizes leveraging existing systems through integration rather than focusing on the development of IMDG. Second, and possibly a secondary contribution is a unique approach to global path planning has been developed. The object to be transported, along with non-prehensile robots, is treated as a single point rigid object for global path planning purposes. This allows for the creation of a comprehensive global path plan, which is then used by individual robots to autonomously identify their local path plans. This method ensures efficient and synchronized movement, leading to successful object transportation. By combining these novel ideas, the research contributes significantly to the field of robotic systems, enhancing both communication and path planning strategies for multi-robot object transportation. A paper summarizing these novel ideas have also been successfully published in the Artificial Life and Robotics Journal by Springer Nature (DOI: 10.1007/s10015-023-00908-5)[39].

1.4 Thesis Outline

The upcoming chapter provides an overview of the relevant literature and previous research conducted in the field. Chapter 3 focuses on the cooperative transport of objects. In Section 3.1, we present the high-level design of the multi-robot system, outlining its architecture and components. Subsequently, in Section 3.2, we detail the control framework specifically developed for the caging and transportation of objects within the system.

Moving forward, Chapter 4 is dedicated to describing the experimental setup. Additionally, we analyze and discuss the results obtained from these experiments, shedding light on the effectiveness and performance of our proposed strategy.

Finally, we provide a comprehensive summary of the research conducted and the key findings. We engage in a discussion regarding potential avenues for future work, highlighting areas that could be further explored and expanded upon to enhance the current approach.

Chapter 2

Literature Review

This section begins with background information on MRS, then moves on to review one of the popular application of MRS, Cooperative Object Transport, and finally reviews the different inter-robot communication mechanisms used by MRS.

2.1 Multi-Robot Systems

Multi-Robot systems (MRS) represent a group of robots operating together within a shared environment and collaborating with each other towards a clearly defined objective [17]. How efficiently the objectives are achieved depends on the robots' capacity to collaborate, and hence it can be perceived as a fundamental characteristic of multi-robot systems [17]. Some of the benefits of multi-robot system are :

- Multiple robots can collaborate and concurrently work on complex tasks to achieve it faster by dividing the work among themselves and increasing the efficiency of the system.

- Based on the capabilities of each robot in the system, various tasks distributed over a wide area can be assigned to the appropriate members to achieve the overall objective.
- With multiple robots for achieving a task provides fault tolerance, making the overall system more robust. Failure of any member of the system can be compensated by others, allowing the system to continue its mission.
- MRS offer scalability and efficiency by allowing the addition or removal of robots based on the task requirements, environmental conditions, or resource constraints.

Based on the above attributes, the MRS can be classified into two main types (A) the collective swarm system and (B) intentionally cooperative systems [43]. A collective swarm system is characterized by the presence of a relatively large number of mobile robots that operate autonomously, following local control laws to achieve coordinated team behavior. These systems exhibit coherent behavior without relying heavily on communication among robots, thereby requiring only minimal communication. An intentionally cooperative system is defined by robots that possess awareness regarding the presence, status, actions, and capabilities of their fellow robots within the team. These robots collaborate and synchronize their efforts to collectively achieve a shared objective. The proposed system would be a intentionally cooperative system since robots share information about themselves with each other by the means of IMDG.

MRS have the potential to enhance various applications. These include, but are not limited to, search and rescue operations [38, 57], forest fire detection [36], hazardous waste removal [56], farm operations [42], mining [66], construction [52], disaster management [35], security applications [35, 22], warehouse management [9], container

handling in harbors and airports [1], as well as games and entertainment, such as soccer [29]. Cooperative object transport is popular application to showcase multi robot system, because it offers a clear domain where both close coordination and cooperation is required to successfully achieve the objective.

2.2 Cooperative Object Transport

Cooperative transportation requires robot teams to move objects from their starting positions to defined goal configurations, sometimes along specified paths. Typically, transportation operates in the plane, and the assumption is made that objects are too heavy or too cumbersome to allow single robots to push alone. Sometimes there are several objects to be moved, with ordering dependencies constraining the sequence of motions.

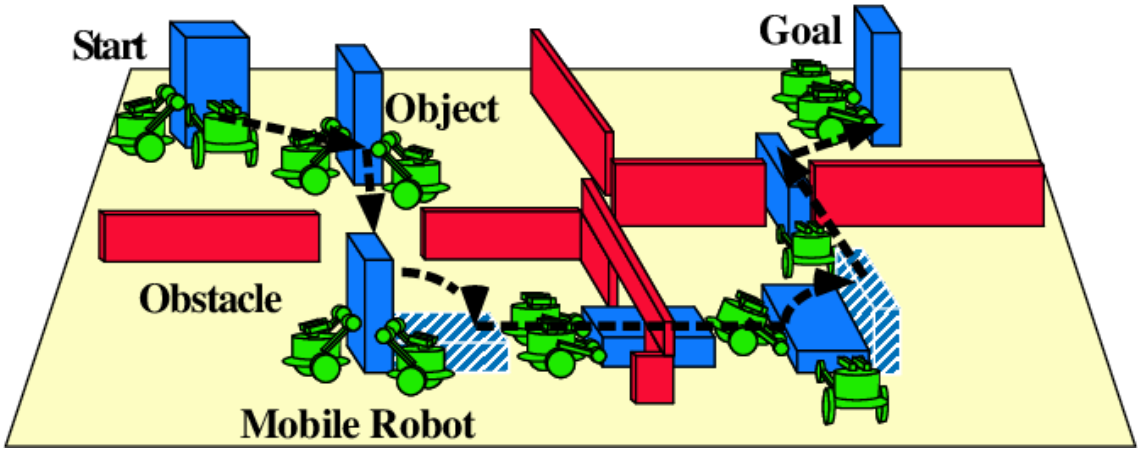


Figure 2.1: Multiple mobile robots engaging in cooperative transportation within a three-dimensional (3D) environment. The robots are tasked with navigating around obstacles, maintaining stable manipulation of the object, and efficiently reaching the goal. [62]

The multi-robot system strategies proposed in the literature for cooperative object

transport can be classified by the interaction mechanisms between the robots and the payload or the type of controller employed [19].

Tuci [55] classify the various approaches based on the interaction mechanisms employed by the robots for handling the object into three categories: (1) Pushing only; (2) Grasping strategy; and (3) Caging Strategy [59, 55]. In the pushing method, transport is achieved by robots pushing the object without being physically attached to it [55, 63, 16, 40]. In the grasping method, robots are physically attached to the object using actuated manipulators that allow them to grasp the object. Transport is then achieved by pushing or pulling (or both) the object while remaining physically attached and maintaining form or force closure conditions throughout the entire process [55, 49, 14, 8]. In the caging method, robots surround the object at a set of fixed positions relative to the object's periphery. Subsequent motion of the robots will also move the object, as long as the cage formation is maintained [55, 45, 50, 50]. Both the caging and pushing methods for object transportation involve robots working collectively without direct attachment to the object. Caging ensures controlled movement with stability, while pushing offers simplicity. A hybrid approach, which combines both methods, is effective for transporting delicate or irregularly shaped objects. Robots form a cage for stability and then use pushing within it for precise movement. The choice between methods depends on the characteristics of the object, with fragile items favoring caging and robust objects using pushing. The overlap highlights multi-robot system versatility, allowing researchers to create adaptable transportation systems catering to diverse needs. If classified based on this approach, the proposed system would be a caging-based cooperative object transport approach. Since the scope of the research was to only focus on non-prehensile manipulation, and to maintain precision of the location of the object being transported, we decided to use the caging strategy. A pushing-only strategy was also explored, but lack of

precision especially near the destination, due to dependence on friction influenced our choice to go with caging rather than pushing.

An alternative approach to classifying and categorizing collective transport methods based on the type of controller and decision-making mechanisms has been proposed in a survey by Farivarnejad et al. [19], the categories of which are: (1) fully centralized controllers; (2) fully decentralized controllers and (3) partially decentralized controllers.The first category includes systems that employ centralized decision-making, focusing on developing strategies for centralized planning, control, and coordination of robots. The fully decentralized approach focuses on swarm or collective based controller by leveraging decentralized coordination strategies, similar to insect swarms. Typically, these systems lack centralized decision making and rely on simple robots capable of exhibiting emergent behavior. Finally, the Partially Decentralized Controllers employ a mixture of centralized and decentralized behavior to different aspects of sensing, computation, and communication. Based on these classifications, the proposed system would be classified as partially decentralized. This is because it provides a balance between, usage of existing fully decentralized features provided by Robot Operating System (ROS) for independent manipulation of the robots, but also sharing of the control between robots when needed by using the IMDG as a shared resource for coordinated transport.

2.3 Inter-robot communication

MRS typically need to be able to share data among their members to efficiently execute their goals. There have been several literature reviews about the different types of mechanisms used in multi-robot system communication. However, classifying

different approaches becomes challenging due to the diverse nature of the attributes of communication.

Yan et al. [64] classify communication methods into (1) explicit and (2) implicit communication based on the information transfer modes. Similarly an alternative approach to categorizing communication methods has been proposed by Farinelli et al. based on how robots exchange information into two categories (1) direct and (2)indirect communication [18]. In direct communication, the members of a MRS send/receive messages to/from each other using a dedicated communication network. Messages are often transmitted using wireless communication protocols. Based on these protocols, message exchange can be between two or selected group members (private), among neighbors within close proximity (local), or among all members (global). In indirect communication, robots are not allowed to communicate with each other explicitly. Instead, they communicate implicitly using the object they transport and/or through the changes in the environment in which they operate. Cao et al. categorize MRS communication according to the mode of interaction: interaction through environments, interaction through sensing, and interaction through communication [6]. Based on these classifications, the proposed system would be categorized to use explicit and direct interaction via communication due to the inherent nature of IMDG.

IMDGs are an existing distributed technology that allows for fast, efficient data access and retrieval, as well as improved scalability, fault tolerance, and data consistency [26]. The use of IMDG technology in robotics is an emerging research area that has the potential to enable efficient and transparent communication, data sharing, and collaboration among robots. The use of IMDG can allow for real-time data processing in robotics due to its ability to store and retrieve data quickly.

An IMDG is a distributed computing technology that stores and processes data in the main memory (RAM) of multiple interconnected nodes that provide a distributed and scalable infrastructure for storing and manipulating data in a highly performant manner. IMDGs are commonly used in various use cases, including real-time analytics, high-speed transaction processing, caching layers for web applications, session management, and distributed caching. They are particularly beneficial for applications that require fast and scalable data processing, such as e-commerce platforms, financial systems, and real-time platforms. IMDGs offer various distributed data structures that allow efficient access and manipulation of data across multiple nodes in the grid. Distributed data structures are a powerful tool that can be used to improve the performance, scalability, and reliability of inter-robot communication [58, 3, 15, 12].

Overall, IMDGs provide a powerful foundation for building distributed systems with the advantages of in-memory storage and efficient data structures tailored for distributed computing.

An alternative to the use of a local IMDG cluster is a cloud-based communication system. The existing literature on cloud-based MRS has focused so far on message buses, queues, and brokers to provide reliable and scalable communication among robots [23, 13, 25]. Applications to tasks such as cooperative object transport have not been studied for cloud-based systems.

Another area of investigation is the security of the data shared by a multi-robot system. Strobel et al. describe a shared knowledge and reputation management system for collective estimation in robot swarms using blockchain technology [51]. They describe how blockchain can be used to provide a decentralized and tamper-proof ledger of robot performance data, which can be used to calculate trust scores.

In this work, we assume that all members of the MRS are trusted. However, the possibility of byzantine robots is useful to consider for future work.

Although use of cloud services will incur cost and use of block-chain can provide security, both of which were out of scope for this research. This work focuses on the use of IMDG established locally among the members of the multi robot system to provide reliable and efficient communication among robot members.

2.4 Path Planning

Path planning plays a fundamental role within a MRS [33]. It serves as a bridge between information perception and motion control, responsible for determining an optimal path for a robot to navigate from a starting point to a goal while avoiding obstacles. It has applications across various domains including manufacturing, logistics, healthcare, and autonomous vehicles. Path planning techniques can categorized into global and local path planning [37].

The global path planning uses environment modeling methods, including grid, topology, geometric feature, and mixed representation methods, along with path evaluation techniques as shown in Figure 2.2. For local path planning, sensors such as laser radar and visual sensors are commonly used.

The algorithms for mobile robot path planning are classified into three categories: classical, bionic, and artificial intelligence algorithms [33]. Classical algorithms encompass cell decomposition, sampling-based methods, graph search algorithms, artificial potential fields, and dynamic window methods. Graph search algorithms such as Dijkstra's algorithm, A* algorithm, and breadth-first search (BFS) have been extensively used for path planning in robotics. These methods focus on exploring the

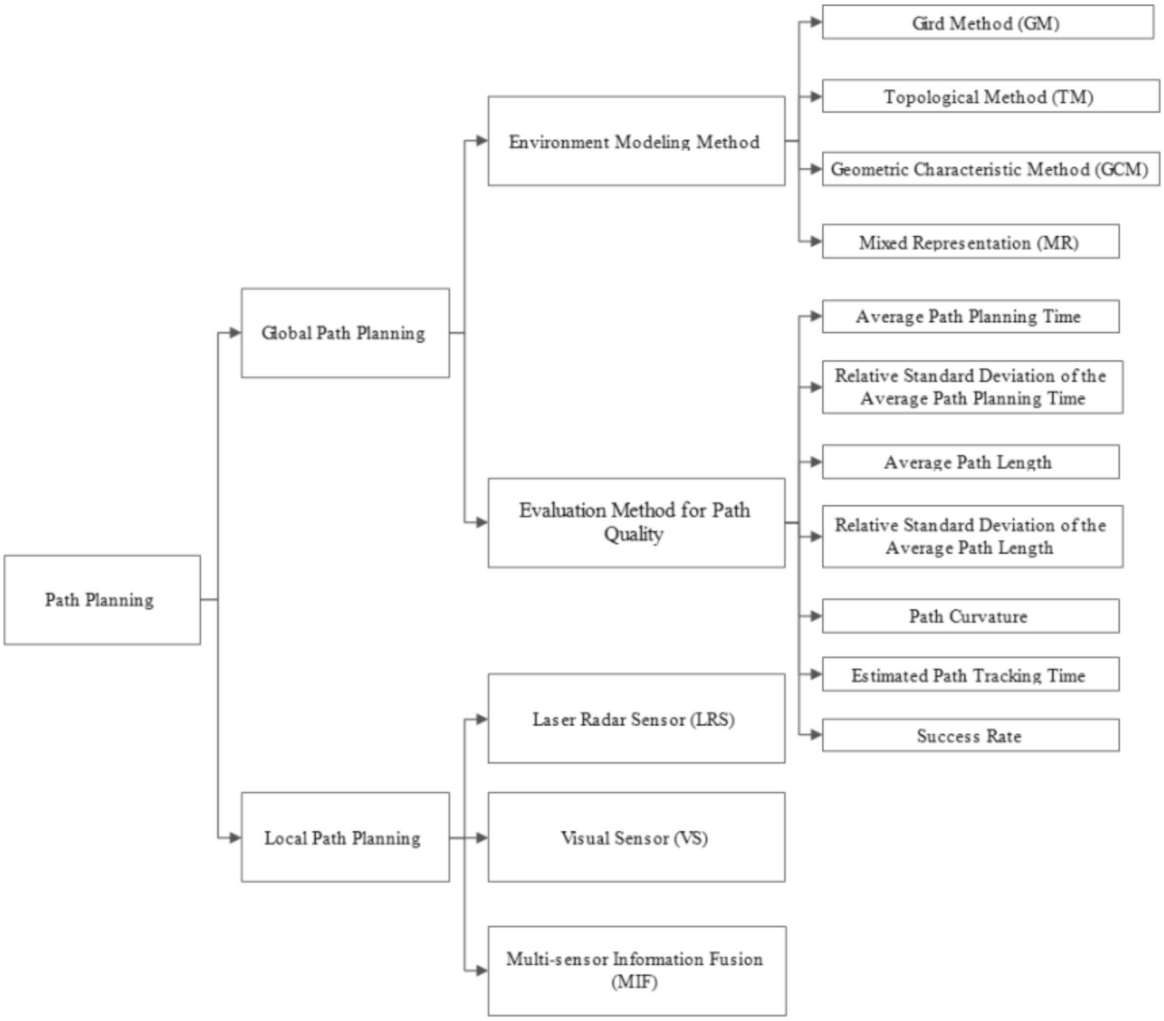


Figure 2.2: Global and local path planning.[33]

configuration space to find the shortest path while ensuring collision avoidance. They are efficient for simple environments but may struggle with complex scenarios due to their exhaustive search nature and lack of consideration for dynamic obstacles. Bio-inspired algorithms include genetic, ant colony, and gray wolf algorithms. Artificial intelligence algorithms cover neural network and fuzzy logic approaches for robot path planning. Reinforcement learning algorithms such as Deep Q-Networks (DQN) and policy gradient methods have been applied to learn navigation policies directly from sensor data [67]. Furthermore, imitation learning techniques enable robots to mimic human demonstrations for path planning tasks [44]. However, challenges remain in terms of generalization to unseen environments and safety guarantees [67].

Since the scope of the current thesis is only limited to static obstacles in a simple environment, with the main objective being able to ensure that robots can coordinately transport object while leveraging IMDG as a means of communication, in this work we only focused on using the global path plan, that used A* algorithm to build a simple path, which is then used by each robots to create their individual path plan.

The scope of the current thesis is limited to static obstacles in a simple environment. Hence the main objective is to ensure that robots can coordinately transport objects while leveraging IMDG as a means of communication. In this work, we focused solely on using the global path plan, which employs the A* algorithm to construct a simple path. Each robot then utilizes this path to create its individual path plan.

Chapter 3

Cooperative object transportation

3.1 High Level Design

A multirobot system is characterized by three key aspects: (1) the types of agents involved, (2) the control architectures employed, and (3) the communication mechanisms utilized [55]. In our proposed control strategy, we have opted for a homogeneous set of robots, which means that all participating robots possess identical physical structures and capabilities. This choice is made because each robot is assigned similar tasks within the system.

Regarding the control architecture, our system adopts a decentralized approach. This means that each robot operates autonomously, making decisions based on its individual perception and local information. The details of this control framework are thoroughly explained in the subsequent section, providing a comprehensive understanding of its design and functionality.

To enable efficient cooperation among robots, a means of communication is essential. This allows the robots to interact, share relevant information, and exchange

data as needed. In our system, we rely on the establishment of an IMDG by leveraging Hazelcast, which itself utilizes TCP/IP/UDP networking protocols. This data grid serves as the communication channel for the indirect exchange of information among robots. By leveraging the IMDG and its underlying networking capabilities, the robots can effectively share their states, coordinate their actions, and collaborate towards achieving the common objectives of the multi-robot system.

3.2 Proposed Control Framework

The proposed control framework consists of two phases: caging and transportation. In the caging phase a sufficient number of robots approach the object, caging it. In the transportation they move it to the desired location in the desired orientation by moving as a group in formation. Each robot moves independently. If there is a disturbance that prevents one robot from moving, the whole system comes to stop, until the obstructed robot can continue again or another robot can substitute them. The flowchart depicting the control framework has been shown in Figure 3.1

Initially (Fig 3.1, subsection a), a supervising system (not covered by this thesis) initiates the control strategy based on an input for the desired state by sending a control message to the most eligible participating individual robots to transport an object from initial to target position. Alternatively the control message can be send by using the ‘rostopic publish‘ command during development and testing. Upon receiving the control message, the members initiate communication with each other by forming a data grid cluster (Fig 3.1, subsection b). For the purposes of this implementation, we leveraged Hazelcast [24] as the middleware providing the data grid system. The usage of this middleware to provide the communication layer, is the primary contribution

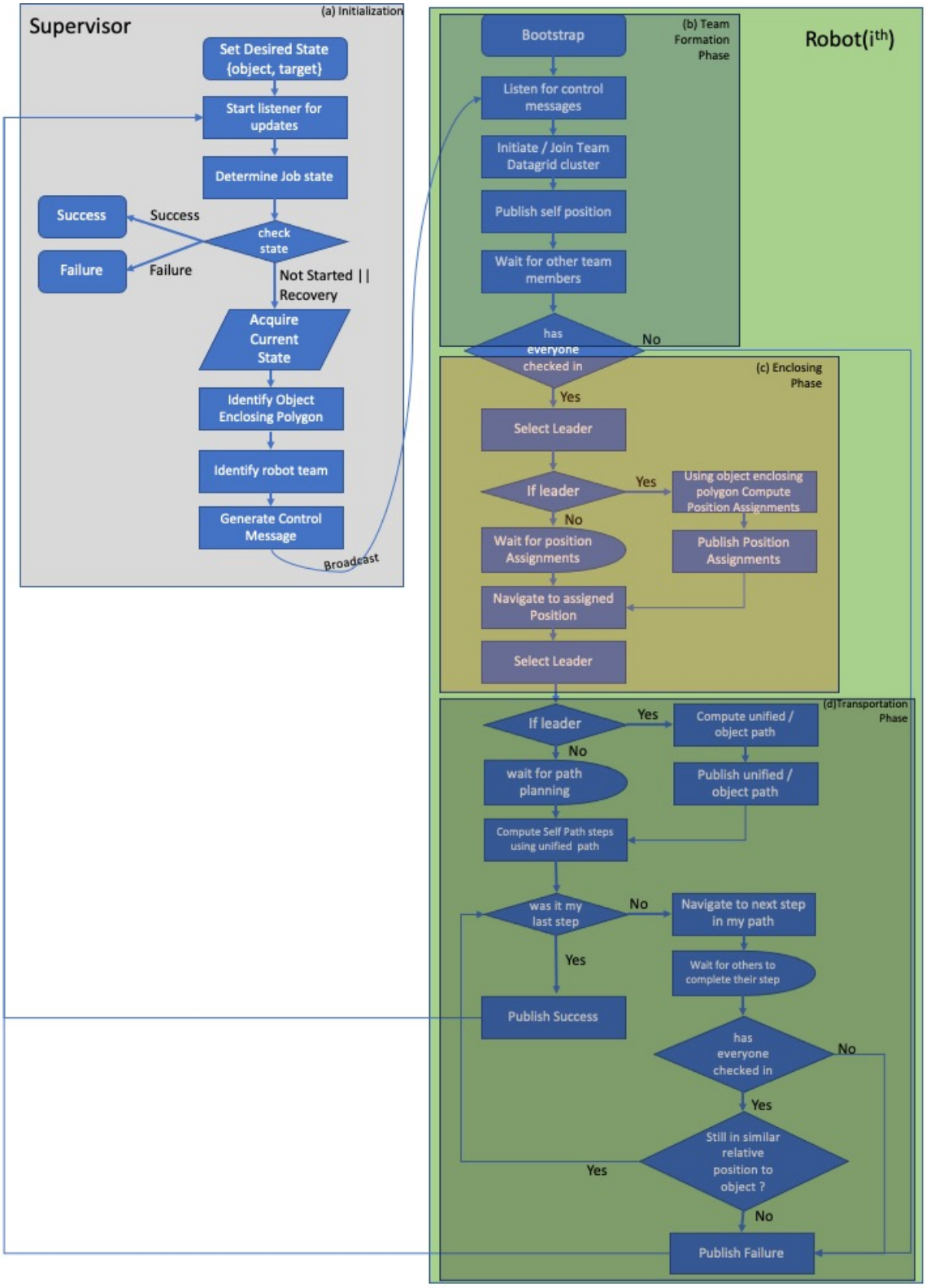


Figure 3.1: The flowchart of the proposed control framework

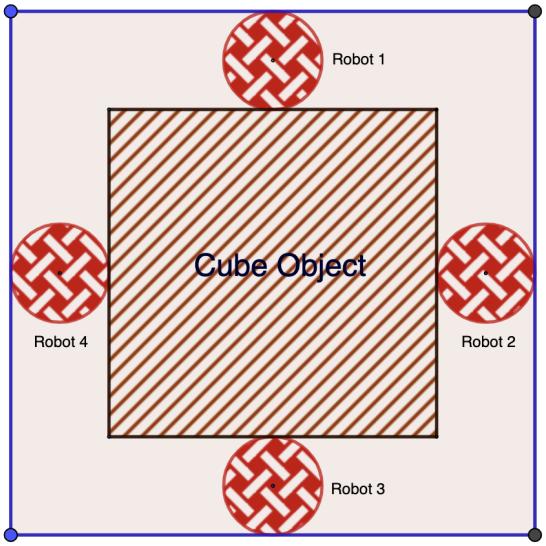


Figure 3.2: Example of bounding polygon around a cubical object and the robots enclosing it. This bounding polygon is considered as a point rigid object for the purpose of global path planning generating a unified path.

of this thesis. The data grid cluster provides distributed data structures like locks, sets, and maps to share information between the robots. Once the cluster has been established by all the members joining the grid, the caging phase is initiated (Fig 3.1, subsection c). If for whatever reason, all the members do not join the cluster within a fixed time interval, other members can inform the supervising system about the failure of the transportation task and the process be can re-initiated, identifying a new set of participating robots.

The next phase of the control strategy is coordinated transportation. During this phase, the members again designate an interim leader, who will be responsible for computing the global path expected to be taken by the object to be transported. To do so, again each member publishes their current position and designate a leader.

In the current implementation, since the leader is only computing the path plan, we have implemented the designation process by using a distributed lock provided

by Hazelcast [24], and designating the path planning as a critical section. A critical section is a code segment that accesses shared variables and has to be executed as an atomic action. The first robot to acquire the lock becomes the leader responsible for generating the path plan for transporting the object. As an alternative, if path planning is to be performed in a distributed manner in the future among all the robots, a leader election process can be held such that then the leader will be responsible for coordinating the sub-tasks of path planning, increasing the efficiency of available compute for such a task.

Now the leader uses the knowledge of the poses of all robots and the object and the object's structure to create a combined bounding polygon for the collective as though it represents a rigid body. The bounding polygon of the object and each of the robots is used to superimpose and compute such a connected representation of the collective. For example, consider Figure 3.2, the robots which are depicted to be circles with red weaving have enclosed a cubical object in the middle, now the information of the object along with the information of the structure of robots, can be used to compute the outer bounding polygon of the collective represented by blue lines.

This representation of the bounding polygon can be considered as a point object and is then used with the map data by appropriately inflating the obstacles in the map for the purposes of path planning. In our implementation, we have used the A* for path planning. Next, once the leader completes the computation of the unified path for the collective, it then publishes it back to all the members. The members now use the unified path and their own relative starting position to the object, to subsequently construct their own individual paths.

Once all robots are finished computing their individual paths, the team initiates the transportation (Fig 3.1, subsection d) of the object by each member navigating

along its own path. In the present implementation while navigating between two points along the path there is no communication among the robots themselves and they navigate autonomously, ensuring they are in the proximity to the same relative position to the object being transported. At every point of inflection along the path, they check in collectively using barrier locking mechanisms to ensure everyone is still in the same relative position to the object, so as to ensure the caging is still in place.

3.3 ROS Implementation

This section discusses the Robot Operating System (ROS)[46] packages that were utilized for various purposes and the implementation of the control strategy in a ROS environment. While the control strategy focused on the logic for transporting the objects, it heavily relies on several features provided by different ROS packages for localization, mapping and navigation.

3.3.1 Localization and Mapping Packages

The robots require to identify their location while working their way through the control strategy to transport the object. To ensure that robots can localize themselves in relation to its environment map, they rely on its on-board sensors. The sensor information is used by packages implementing a process called Simultaneous Localization and Mapping (SLAM) to provide the respective robots their location details .

The most popular SLAM packages available in the ROS framework are hector_mapping and gmapping. Additionally, the developers of the robot model we planned to use for simulation (more discussed in next chapter) also provide a package for the purposes of SLAM called neo_localization.

The hector_mapping [30] package is a SLAM approach suitable for platforms that exhibit roll/pitch motion, eliminating the need for odometry. It leverages 2D pose estimates from LiDAR sensors. While the system lacks explicit loop closing ability, it remains effective for real-world scenarios. In particular, the system has found successful applications in diverse projects involving unmanned ground robots.

The ROS package slam_gmapping [21] serves as a wrapper for OpenSlam's Gmapping, enabling its integration into Robot Operating System (ROS) environments. To use slam_gmapping, a mobile robot is essential, providing odometry data and equipped with a horizontally-mounted, fixed laser range-finder. The slam_gmapping node processes laser and pose data collected by the mobile robot to generate a 2D occupancy grid map, resembling a building floor plan. It also attempts to transform each incoming laser scan into the odometry (odom) tf frame for accurate mapping and localization. In ROS, the 'tf' package is an integrated tool that enables users to effectively monitor multiple coordinate frames over a period of time. 'tf' manages the connections between coordinate frames within a time-buffered tree structure, allowing users to seamlessly convert points, vectors, and other entities between any two coordinate frames at any specified moment in time [47].

The neo_localization [41] package offers a straightforward yet highly effective alternative to the standard Adaptive Monte Carlo Localization (AMCL) method. Similarly to a particle filter, it generates particles from scratch during each update, and each particle undergoes multiple Gauss-Newton iterations for optimization. The primary objective of this package is to achieve improved accuracy and robustness, especially in challenging environments. Remarkably, depending on the map quality, localization accuracy of up to 1 mm can be attained.

Moreover, the neo_localization package exhibits intelligent behavior by automatically transitioning into a constrained 1D mode or even 0D mode (relying solely on odometry) when surroundings are inadequate for localization. Notably, it comes preinstalled on Neobotix platforms since April 2020, providing users with a powerful localization solution for their robots.

3.3.2 Navigation Packages

The robots need a navigation system to be autonomous. Currently autonomous navigation is used by the robots when approaching the object for caging. The basic autonomous navigation system in the ROS environment is performed using the move_base [34] package. As seen in Figure 3.3, the SLAM process data are inputted into the move_base package along with the AMCL information and the map of an area. The move_base package provides an implementation of an action for the robot, where a goal pose is defined and travel is attempted. The move_base package contains a node that provides a ROS interface for configuring, running, and interacting with the navigation stack on a robot. The move_base node utilizes two planners to perform the navigation task: global and local. To ensure that obstacles are identified, the node maintains two costmaps: one for the global planner and one for the local planner. Figure 3.3 presents a high-level view of the move_base node. The processed data in the move_base package which publishes cmd_vel, the velocity, to the robot's controller and the visualization of the navigation system in 3D visualization tool for ROS call 'rviz'.

The robots need a navigation system to make them autonomous, and within the ROS environment, the move_base package serves as the fundamental autonomous navigation system. As shown in Figure 3.3, this system integrates data from SLAM,

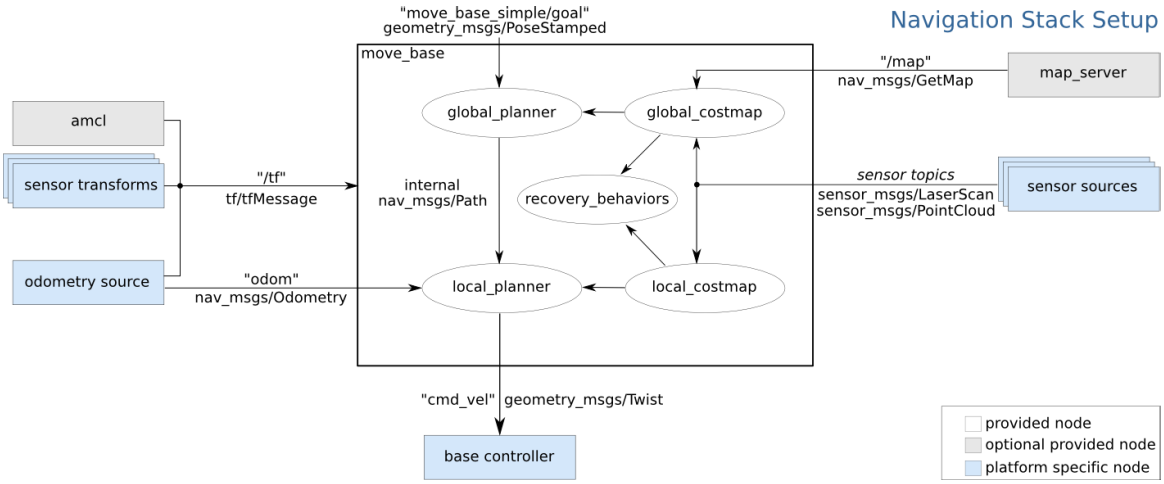


Figure 3.3: A High-level Schematic of the move_base Node [34]

AMCL and a map of the area. The move_base package enables the robot to define and attempt travel to a goal pose through an implemented action. Its node acts as a ROS interface, facilitating configuration, execution, and interaction with the navigation stack on the robot.

The move_base node employs two planners, global and local, to accomplish the navigation task effectively. To accurately detect obstacles, two cost maps are maintained: one for the global planner and another for the local planner. A concise overview of the move_base node is depicted in Figure 3.3. The processed data in the move_base package results in the publication of cmd_vel (velocity) to the robot's controller, enabling smooth and precise movement, while the navigation system's visualization can be observed in rviz.

The move_base node utilizes three plugins to enable obstacle detection during basic autonomous navigation. The static map layer furnishes unchanging data to the global planner, utilizing known map information to identify obstacles. Meanwhile, the local planner relies on the obstacle layer and the inflation layer. The obstacle layer uses data from lidar sensors' point clouds to detect both static and dynamic obstacles.

On the other hand, the inflation layer ensures smooth robot traversal through the environment, preventing it from getting stuck.

The local planner creates a local map of the environment and, if it detects dynamic objects, such as people, it inflates the obstacle to the size of the inflation radius. The robot's path is then redirected to avoid the inflated obstacle while navigating around it. This combined approach of the three plugins for global and local planners allows the robot to autonomously navigate and approach the target object while successfully adopting the caging position.

3.3.3 Control Framework

The pseudo code for the implementation of the overall control strategy is detailed in Algorithm 1. While in the previous section 3.2 we discussed about two high-level phases of enclosure and transport, during implementing an additional phase can be added in the beginning. The reason being that each of them represents a unit at the end of which all robots synchronize with each other about having successfully completed the task at hand. They make use of the countdown lock provided by IMDG and listen for the countdown counter. If for any reason any robot does not count down itself, within a preset timeout, the operation is considered having failed.

In ROS, the control framework is implemented by an action server node interface provided by the actionlib package that listens for "Transport" goal message. Upon receiving the goal request, the strategy is triggered. The first phase is to establish the team, each robot starts its own Hazelcast client and waits for other members to join. Only if all the members have joined, then the strategy proceeds or it returns an action result indicating failure.

The next phase is the enclosing phase, where in one of the robots will obtain a distributed lock, compute the optimal caging positions for each of the robots, and share them back via the use of a distributed map. Each of the robots read their assigned positions from the map and begin autonomous navigation using the move_base package. As described in the section 3.3.2 move_base in itself is another package that implements an action server, the robot sends the assigned positions as a goal to the move_base package and waits for a result from move_base. Once the result from the move_base package is received, each robot verifies its current position is indeed the assigned position, and again waits for all other robots using a countdown lock to affirm that all others have reached their assigned position as well. Again only if all other robots have reached their assigned position within a preset time period, then the strategy proceeds into the transportation phase.

Finally the transportation phase begins by one of the team member obtaining a distributed lock. The one to obtain the lock computes the unified global path plan by considering the object and the robots as a connected rigid body. To do so it combines them by computing the union of the bounding boxes of the robots themselves and the object. The final generated polygon is used along with the map provided by the AMCL package as an input to the A* search to identify the global path plan. The path is shared with all other robots through the use of a distributed list. Each of the robots read the global plan, and generate their own plan by attempting to keep the same relative position to the object. Once the local plan has been identified, the points of inflection along the local plan are identified. Inflection points are points where the trajectory changes concavity, i.e. from being "concave up" to being "concave down" or vice versa [60]. These points of inflection serve as checkpoints where every robots checks in with each other by the means of counter to ensure they are still in the same relative position to the object, and have successfully made to the checkpoint.

If all of them have successfully reached the checkpoint, they continue the transport until the final destination or checkpoint is reached. All the robots reaching their final checkpoint while still maintaining the same relative position to the object will imply that the transportation of the object is complete. At this point, they communicate the result back to the supervisory system about the status of the operation by sending back the action server result message, completing the task. The pseudocode for the implementation of the overall control strategy is detailed in Algorithm 1 .

Algorithm 1: Control Framework

Phase 1: Team Formation

Data: $teamMembers(m[n]), objectEnclosingPolygon$
$(x_o, y_o, \theta_o, v[n]), targetPose(x_o^d, y_d^o)$
Result: IMDG Cluster formation between the team_members
→ Wait for all team members to join the cluster, if not fail

Phase 2: Enclosing Phase

Data: $teamMembers(m[n]), objectEnclosingPolygon$
$(x_o, y_o, \theta_o, v[n]), targetPose(x_o^d, y_d^o)$
Result: $x_i^e(t), y_i^e(t)$
→ Compute the position assignment for each robot $(x_i^e(t), y_i^e(t))$;
→ Each robot publishes request to its move_base action server (ROS Nav package);
while *not at assigned position* **do**
    evaluate Δx_i and Δy_i;
    monitor Δx_i and Δy_i;
end
→ send confirmation of enclosing phase completion → wait for others to reach the same state, if not fail

Phase 3: Transportation Phase

Data: $x_o, y_o, x_i^e(t), y_i^e(t), x_o^d, y_o^d$
Result: $x_i^d(t), y_i^d(t)$
→ Compute the unified global path plan by considering the object and robots as continuous rigid object and publish the global plan;
→ Each robot computes its individual path plan, by maintaining the same relative position to the center of object which tracks along the global path plan;
while *in transportation phase* **do**
    evaluate $\Delta x_i = x_i$ - $x_i^d(t)$ and $\Delta y_i = y_i$ - $y_i^d(t)$;
    if $\Delta x_i^2 + \Delta y_i^2 > \epsilon$ **then**
        compute v_i, ω_i;
    else
        $v_i = 0$;
        $\omega_i = 0$;
    end
    ensure the current relative distance is same as original
end
→ wait for others to reach the same state, if not fail;
→ send confirmation of transportation phase completion;

Chapter 4

Simulation

4.1 Experimental Setup

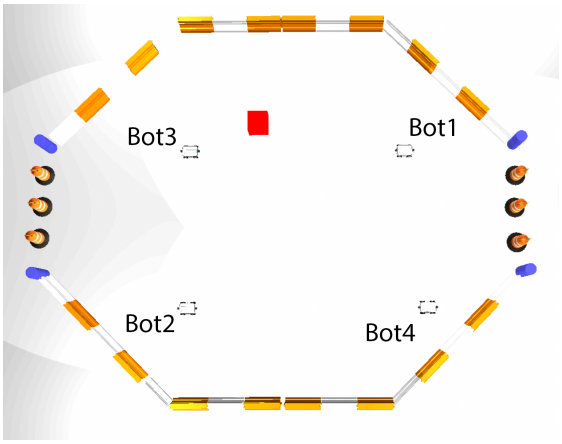
To test and validate the proposed cooperative control strategy, ROS middleware suite and the Gazebo simulator were used. For the robots, we have leveraged the models of Mobile Robot MPO-500 by NeoBotix Robots, which have mecanum type of wheels (omni directional wheels) and hence can provide exceptional maneuverability and the ability to move smoothly in any direction. Based on the shape of the robots, adjustments in orientation might be needed when caging an object. Since the MPO 500 are rectangular, we orient them parallel to the side of the object they are enclosing. The stage for the simulation is set in a Gazebo world provided by NeoBotix Robots called the neo track. It is a simple world, with a boundary surrounded by construction barrels and angled barriers. The robot and the object to be transported are initiated at random positions prior to starting the simulation using ROS launch files. These positions can also be modified once the Gazebo interface has started.

4.2 Simulation Results

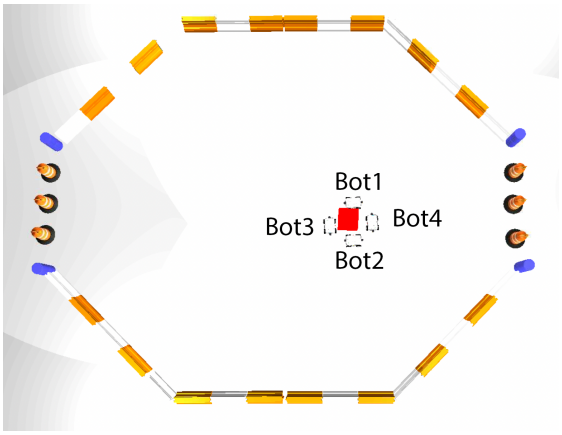
The first simulation experiment is conducted in an open environment with no obstacles. Thereafter we start adding different obstacles, in order to better represent real-world conditions. Screenshots were captured at the beginning of the simulation's run, during transit, and at the end. These screenshots are shown in sub-figures (a)-(c) of Figures 4.1, 4.2, 4.3, and 4.4 of the trials described below. Also, the corresponding plots for the trajectories taken by the robots and the error signals along with the resulting trajectories have been captured in subfigures (d) and (e) respectively of Figures 4.1, 4.2, 4.3, and 4.4. The errors shown in each sub-figure (e) are the distances between the planned positions along the path for each robot/obstacle and the actual positions.

4.2.1 Trial with no obstacles

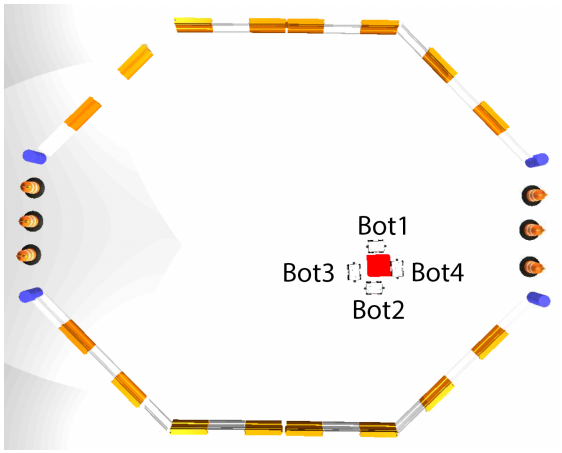
In this trial there are no obstacles, and the robots have to transport a object from one point to another. Sub-figures (a) - (c) of Fig. 4.1 show that the robots were successfully able to transport the object. The subfigure (d) of Fig. 4.1 shows the trajectories taken by the robots, and subfigure (e) of Fig. 4.1 shows that the error in position for both robots and the object stays at low levels.



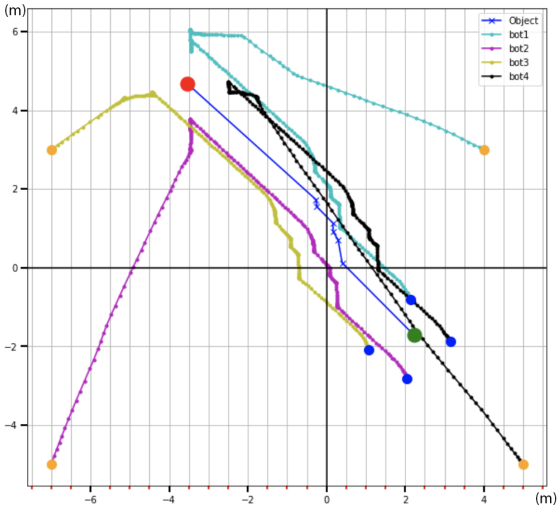
(a) Initial State



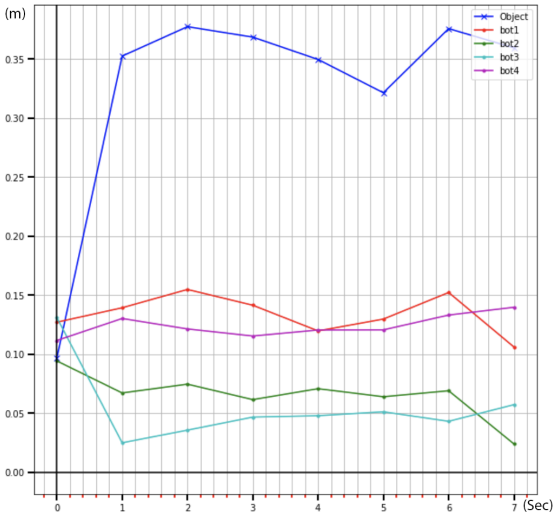
(b) Object in transit



(c) Transportation Completed



(d) Trajectories



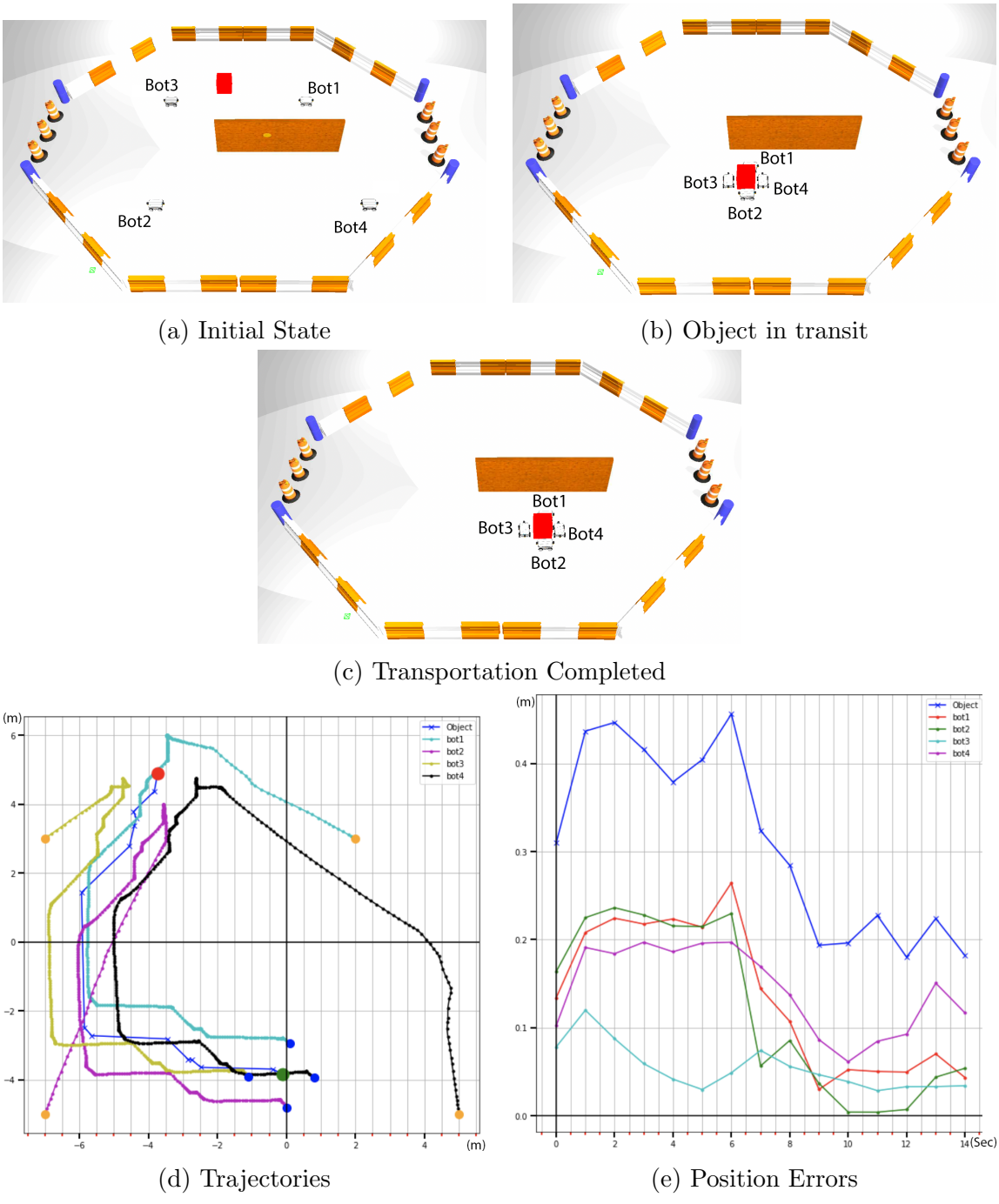
(e) Position Errors

Figure 4.1: Results in the open environment. **(a)-(c)** show screenshots captured during the initial, in-transit and completed states. The red cube is the object to be transported. **(d)** shows the trajectories of all robots and the object. The object's start and final positions are shown as large red and green circles, respectively. **(e)** shows errors in position over time.

4.2.2 Trial with horizontal static obstacle

During this trial, we introduced a horizontal wall into the environment, creating an obstruction that prevented the robots from directly transporting the object in a top-to-bottom direction. Sub-figures (a) - (c) of 4.2 show that the robots were successfully able to transport the object to the other side of the wall. Sub-figure (d) of Figure 4.2 provides a visual representation of the trajectories followed by the robots during the object transport process.

Sub-figure (e) of Figure 4.2, illustrates the observed positional errors. It is noteworthy that these errors stabilize after a while. This stabilization demonstrates the robots' ability to maintain accuracy and consistency in object transport, ensuring that the object reaches its destination as intended.



(a) Initial State

(b) Object in transit

(c) Transportation Completed

(d) Trajectories

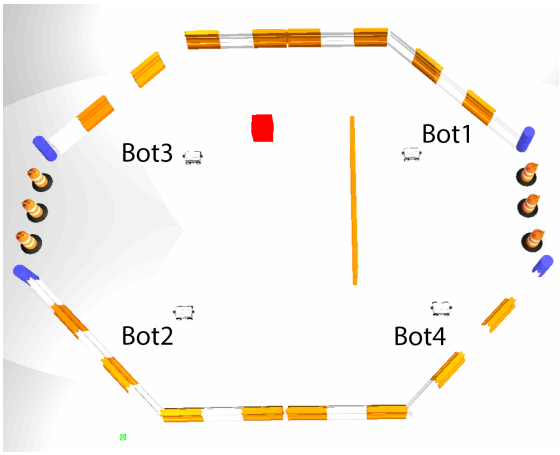
(e) Position Errors

Figure 4.2: The environment has a horizontal wall and the robots have to transport the cube from top to bottom. See Fig. 4.1 for interpretation.

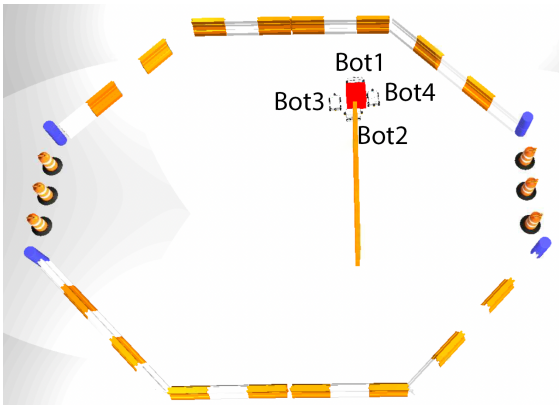
4.2.3 Trial with vertical static obstacle

In this particular test, we introduced a vertical wall into the environment, creating an obstacle that hindered the direct transport of the object from the left side to the right side. As seen in the screenshots of subfigures (a) - (c) of 4.3, it is evident that the robots succeeded in overcoming this challenge and performing the task of transporting objects. In particular, the robots navigated in close proximity to the wall. It is worth noting that the ability of the robots to pass close to the wall is influenced by the configured inflation radius for the obstacles. Increasing the inflation radius would generate a path that deviates further away from the wall. This parameter allows for flexibility in determining the desired proximity of the robots to the obstacle during object transport, providing control over the safety margin based on the specific requirements of the task.

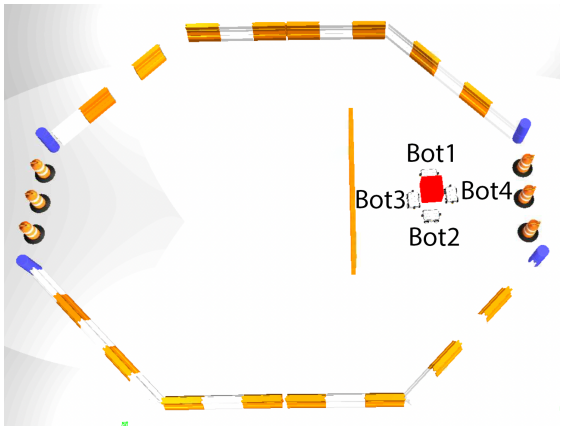
The trajectories displayed in sub-figure (d) of Figure 4.3 provide a visualization of the planned path generated using the A* search algorithm. This path planning technique enables the robots to navigate efficiently around the vertical wall, finding an optimal route that circumvents the obstacle and ensures successful object transport. The positional errors shown in Figure 4.3(e) demonstrate consistently low values throughout the entire object transport process. This signifies the accuracy and precision achieved by the robots during the execution of the task, minimizing deviations between the expected and actual positions of both the object and the robots.



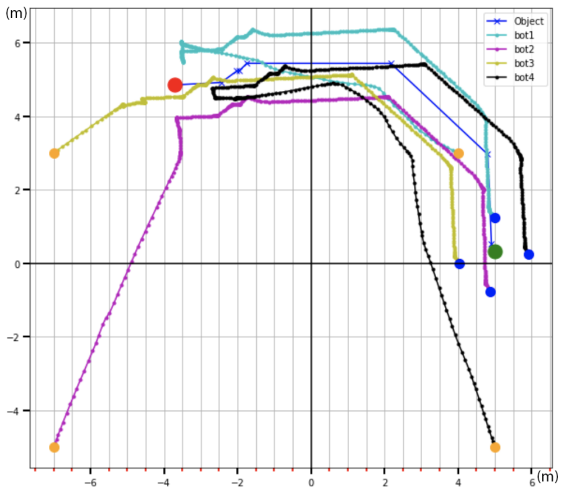
(a) Initial State



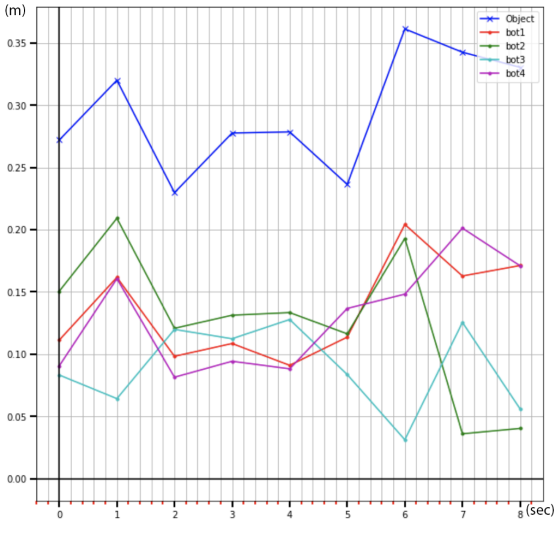
(b) Object in transit



(c) Transportation Completed



(d) Trajectories

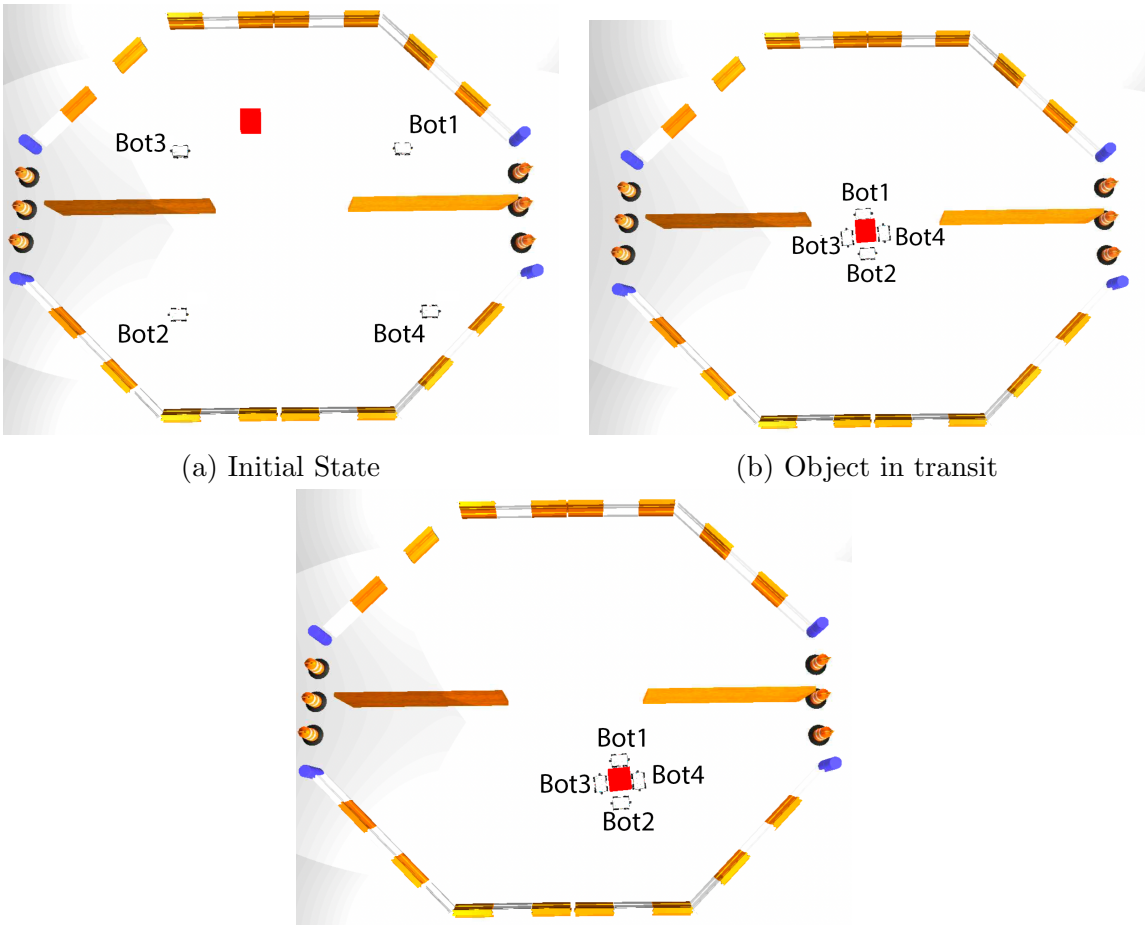


(e) Position Errors

Figure 4.3: The environment has a vertical wall and the robots have to transport the cube from left to right. See Fig. 4.1 for interpretation.

4.2.4 Trial with 2 horizontal static obstacles depicting a passage

During this trial, we introduced two horizontal walls into the environment, strategically positioning them so that the object was transported to the other side by navigating the gap between them. Importantly, the passage width was carefully set to accommodate the entire configuration of the robots and the object. By examining the screenshots presented in subfigures (a) - (c) of Fig. 4.4, it is evident that the robots effectively accomplished the task of transporting the object, demonstrating their ability to navigate through the designated gap between the walls. The successful completion of this trial underscores the efficiency and capability of robots to maneuver the object through challenging spatial constraints.



(a) Initial State

(b) Object in transit

(c) Transportation Completed

(d) Trajectories

(e) Position Errors

Figure 4.4: The environment has two horizontal walls and the robots have to transport the cube through the passage between them. See Fig. 4.1 for interpretation.

4.3 Observations from deliberate disruptions in experimental trials

In the previous section, we examined the successful trials. Now, in this section, we will focus on the trials deliberately disrupted by introducing various challenges. Please note that this is not an exhaustive list, but rather an exploration of the challenges that could be countered and possibly how they could be handled if the system were deployed in the real world.

4.3.1 Network Failures

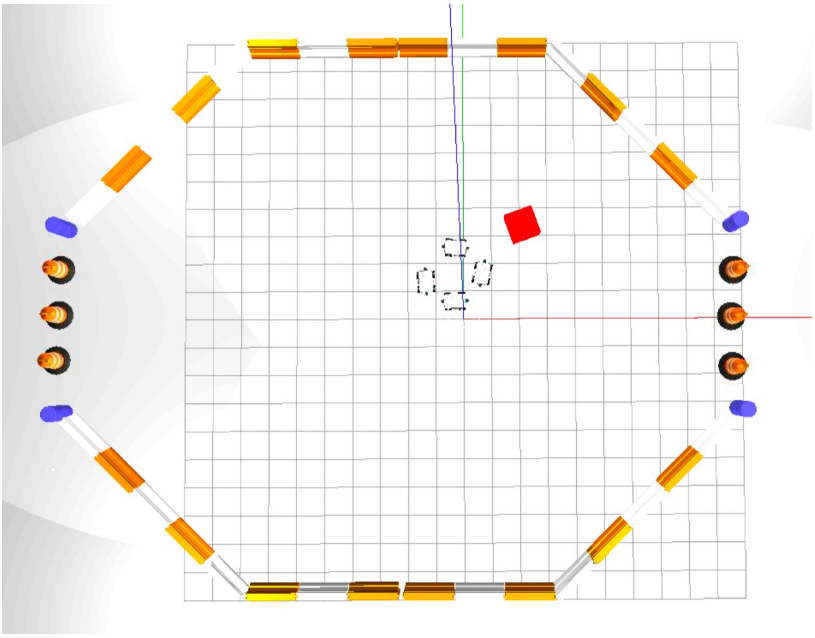
Network failures represent a critical challenge that can prohibit the seamless exchange of data between robots since this will break the cluster formed by the IMDG. These interruptions can arise from several factors, such as signal interference, radio frequency (RF) congestion, communication channel errors, hardware malfunctions, or environmental obstacles [2]. In general when such failures occur, the consequences can range from temporary interruptions in operations to complete breakdowns in communication, compromising the overall efficiency and reliability of the system.

In our current implementation, in such scenarios, it was observed that the robots continue to perform their current operation, until a checkpoint is reached, at which point, since all robots wait to hear about each others completion by the use of distributed countdown lock provided by IMDG. Since the countdown lock will not reach zero and timeout, the robots realize, for some reason, that they are unable to communicate with other robots or that others haven't completed their tasks. Currently there are no recovery methods after the timeouts and scheduling supervisor will be notified about the failure of the operation.

4.3.2 Robot failure or broken cage formation

Physical robot failures encompass a wide range of mechanical, electrical, and structural malfunctions that can compromise the operational integrity of a wireless robotic system. These failures can manifest in components such as actuators, sensors, communication modules, power systems, or even the robot's mechanical body itself [7]. When a robot experiences failure, it can result in unexpected and potentially hazardous behavior, leading to mission-critical disruptions, loss of productivity, or, in some cases, even safety risks for both the robot and its surrounding environment.

One such scenario critical to a transportation system using the caging approach is the breaking of the cage and the object being transported escaping the cage 4.5. Similar to network failures, in the case of broken cages or robot failures in general, it has been noted that the robots persist in executing their ongoing operations until they reach next checkpoint. At this juncture, all robots enter a waiting state, actively listening for updates on each other's completion through the implementation of a distributed countdown lock facilitated by an IMDG. However, since the robots will not be in the same relative position to the object, the countdown lock does not reach zero and time out, leading to robots to recognize that for some reason, communication with other robots is not possible or that other robots have not completed their tasks. At present, there are no recovery methods in place following these timeouts. The scheduling supervisor will be promptly notified about the failure of the operation once the timeout occurs.



(a) Broken Cage Scenario

```
[ERROR] [1691671624.994553, 2840.537000]: Detectected broken cage error, terminated
Traceback (most recent call last):
  File "/home/priyanknarvekar/workspaces/ros/melodic/src/accio_action_server/src/ActionServer.py", line 74, in execute_cb
    self.transport(team_name, team_hz_address, team_hz_pass, team_members, objectEnclosingPolygon, target_pose)
  File "/home/priyanknarvekar/workspaces/ros/melodic/src/accio_action_server/src/ActionServer.py", line 324, in transport
    raise Exception("Detectected broken cage error, terminated")
Exception: Detectected broken cage error, terminated
[DEBUG] [1691671624.998010, 2840.537000]: Transitioning to WAITING_FOR_RESULT (from ACTIVE, goal: /bot3/accio_cc_node-1-2744.667)
[DEBUG] [1691671625.009380, 2840.547000]: Transitioning to DONE (from WAITING_FOR_RESULT, goal: /bot3/accio_cc_node-1-2744.667)
[INFO] [1691671625.021653, 2840.557000]: Error Signal Info from /bot4/ : (672665820445749/1000000000000000, 32385159829473/20000000
0000000) | (1.2476999411636172, 3.321407842311773) | (-0.4, 1.4000000000000001)
[INFO] [1691671625.032525, 2840.567000]: current_relative_postion : Point2D(-0.575034120717868, -1.70214985083812)
[INFO] [1691671625.040501, 2840.577000]: /bot4/ : Detectected broken cage, breaking out of the loop
[DEBUG] [1691671625.047048, 2840.587000]: Accio Transport result result:
  data: False
[DEBUG] [1691671625.058108, 2840.597000]: Sending HZ message: {'from': '/bot3/accio_cc_node', 'result': False}
[ERROR] [1691671625.060027, 2840.597000]: Detectected broken cage error, terminated
Traceback (most recent call last):
  File "/home/priyanknarvekar/workspaces/ros/melodic/src/accio_action_server/src/ActionServer.py", line 74, in execute_cb
    self.transport(team_name, team_hz_address, team_hz_pass, team_members, objectEnclosingPolygon, target_pose)
  File "/home/priyanknarvekar/workspaces/ros/melodic/src/accio_action_server/src/ActionServer.py", line 324, in transport
    raise Exception("Detectected broken cage error, terminated")
Exception: Detectected broken cage error, terminated
```

(b) Robot stdout log messages for the detected broken cage

```
priyanknarvekar@priyank-dev$ ./src/control_center.py cube_r_1m
{'target_position': {'y': -2.0, 'x': 2.0, 'z': 0.0}, 'team_hz_address': '10.171.189.217:5702', 'team_members': ['bot1', 'bot2', 'bot
3', 'bot4'], 'from': 'control_center', 'polygon': [{'y': 78253/20000, 'x': -37519/12500, 'z': 0}, {'y': 98253/20000, 'x': -37519/125
00, 'z': 0}, {'y': 98253/20000, 'x': -50019/12500, 'z': 0}, {'y': 78253/20000, 'x': -50019/12500, 'z': 0}], 'team_hz_pass': 'passwor
d', 'team_name': 'task1', 'target_orientation': {'y': 0.0, 'x': 0.0, 'z': 0.0, 'w': 1.0}}


Got message: {'from': '/bot2/accio_cc_node', 'result': False}
Publish time: 1691671624985
```

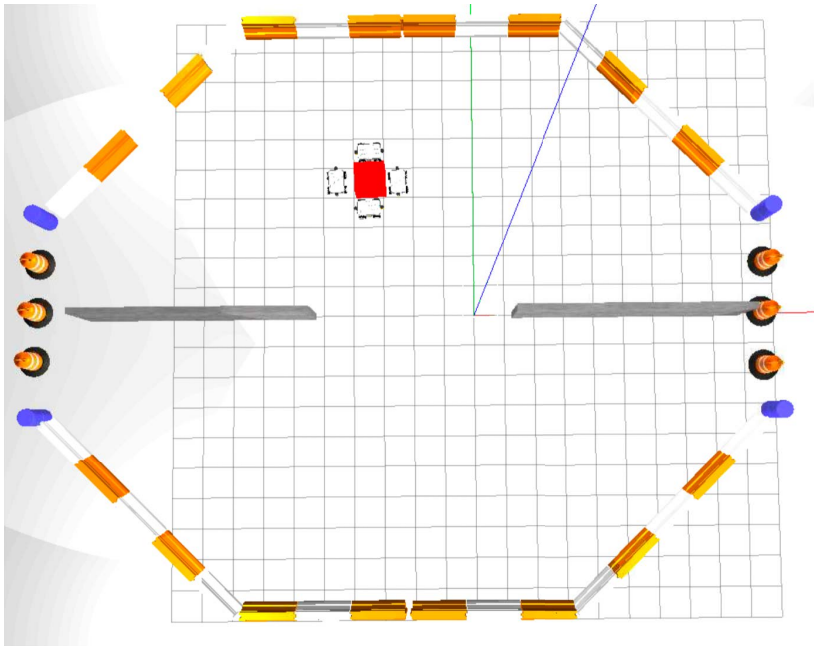
(c) Supervisor system receiving the message that task failed

Figure 4.5: Situation showing a broken cage scenario, which was induced intentionally. The figures show that the robots detected such scenario, stopped the transportation, and passed the result back to the supervisory system.

4.3.3 Narrow Passages

Narrow passages present a unique obstacle for robotic formations, as they require precise coordination and adaptability to traverse safely and efficiently. These passages can be found in various real-world scenarios, such as confined spaces in disaster-stricken areas, cluttered industrial settings, or constrained urban environments. When navigating through such passages, robotic formations must overcome spatial limitations, potential collisions with surrounding obstacles, and the risk of formation breakdown due to communication disruptions.

The narrow passages which cannot accommodate the robots formation along with object, the current implementation fails when trying to identify a path plan. This is because the robots, along with the object, are currently treated as a single, rigid object 4.6. The team will not be able to calculate a global path plan and notify the supervisory system about failure of operation.



(a) Narrow path Scenario induced by setting high inflation radius

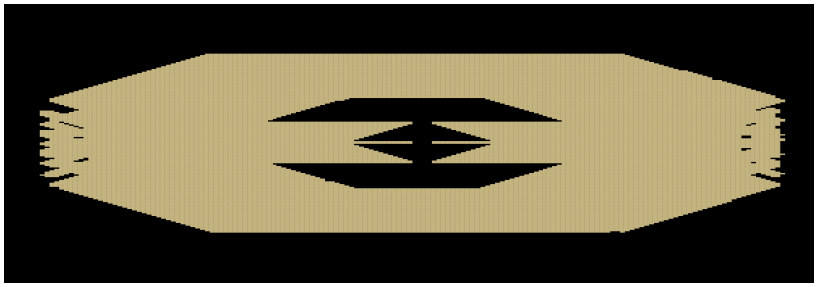
```
[INFO] [1691673817.909407, 5064.467000]: /bot1/ : Converting map response from map server to 2d map
[INFO] [1691673831.431274, 5077.947000]: /bot1/ : Converted map response from map server to 2d map
[INFO] [1691673831.434462, 5077.947000]: /bot1/ : Downsampling the map
[INFO] [1691673833.164172, 5079.617000]: /bot1/ : Downsampling complete
[INFO] [1691673833.176344, 5079.647000]: /bot1/ : Identifying obstacles for inflation
[INFO] [1691673835.037728, 5081.507000]: /bot1/ : Creating Grid
[INFO] [1691673841.114072, 5087.557000]: /bot1/ : Converting world coords(-43769/12500,88253/20000) to map coord
[INFO] [1691673841.122959, 5087.567000]: /bot1/ : Resulting map coord(482,477)
[INFO] [1691673841.125600, 5087.567000]: /bot1/ : Converting world coords(2.0,-2.0) to map coord
[INFO] [1691673841.127346, 5087.567000]: /bot1/ : Resulting map coord(510,510)
[INFO] [1691673841.273526, 5087.717000]: /bot1/ : Generated Grid path => []
[ERROR] [1691673841.384143, 5087.747000]: Could not identify Path
Traceback (most recent call last):
  File "/home/priyanknarvekar/workspaces/ros/melodic/src/accio_action_server/src/ActionServer.py", line 74, in execute_cb
    self.transport(team_name, team_hz_address, team_hz_pass, team_members, objectEnclosingPolygon, target_pose)
  File "/home/priyanknarvekar/workspaces/ros/melodic/src/accio_action_server/src/ActionServer.py", line 283, in transport
    path = self.getRealPath(objectEnclosingPolygon.centroid,target_pose.position)
  File "/home/priyanknarvekar/workspaces/ros/melodic/src/accio_action_server/src/ActionServer.py", line 411, in getRealPath
    raise Exception("Could not identify Path")
Exception: Could not identify Path
```

(b) Robot stdout log messages for the detected broken cage

```
priyanknarvekar@priyank-dev$ ./src/control_center.py cube_r_1m
{'target_position': {'y': -2.0, 'x': 2.0, 'z': 0.0}, 'team_hz_address': '10.171.189.217:5702', 'team_members': ['bot1', 'bot2', 'bot
3', 'bot4'], 'from': 'control_center', 'polygon': [{'y': 78253/20000, 'x': -37519/12500, 'z': 0}, {'y': 98253/20000, 'x': -37519/125
00, 'z': 0}, {'y': 98253/20000, 'x': -50019/12500, 'z': 0}, {'y': 78253/20000, 'x': -50019/12500, 'z': 0}], 'team_hz_pass': 'passwor
d', 'team_name': 'task1', 'target_orientation': {'y': 0.0, 'x': 0.0, 'z': 0.0, 'w': 1.0}}


Got message: {'from': '/bot1/accio_cc_node', 'result': False}
Publish time: 1691673841366
```

(c) Supervisor system receiving the message that task failed



(d) Visualization depicting the highly inflated map (radius=15) resulting in narrow passages, such that a path could not be found.

Figure 4.6: The environment has two horizontal walls and the robots have to transport the cube through the passage between them. However since the inflation radius was set too high, no path could be identified. This results in the task failing and notifying the supervisory system as such.

4.4 Discussion of Results

We have demonstrated successful object transport in an open environment and in the presence of obstacles in three different configurations. In all the different scenarios, the system demonstrated effective operation, achieving the desired objective of transporting the object to the goal position. However, it is important to acknowledge certain limitations that were encountered during the study.

Positional error sub-figures (e) of figures 4.1, 4.2, 4.3, and 4.4 indicate the positional accuracy of the transported object and the robots in relation to the planned path. These subfigures illustrate the discrepancy between the expected or planned and actual positions during the object transportation process. However, it should be noted that these errors stabilize after the initiation of transportation. To achieve further reductions in positional errors, it would be beneficial to explore the influence of cage tightness and relaxation on the transportation process. This avenue of investigation holds promise for enhancing the precision and accuracy of object transport if minimized positional errors are a critical requirement.

One notable limitation is the system's inability to identify suitable paths in narrow passages. In such cases, the robots are unable to create a path plan and would require some sort of reconfiguration of their caging positions, which is currently unsupported by the system. This constraint highlights an area for future development and improvement, as enabling reconfiguration in response to narrow passages would enhance the system's versatility and adaptability in diverse environments.

Overall, the simulations have demonstrated the successful transport of objects in an open environment while considering various configurations and obstacles. Although the system performed well overall, limitations such as the handling of narrow passages

and minor positional errors were identified. Future research efforts should focus on addressing these limitations, particularly by enabling reconfiguration in narrow passages and exploring methods to reduce positional errors based on cage tightness and relaxation.

Chapter 5

Conclusions

Through various simulations and evaluations, we have demonstrated the effectiveness of our approach. The coordinated control strategy successfully facilitated the transportation of objects, achieving the desired goal of relocating them to their intended targets. This outcome highlights the robustness and reliability of our proposed solution.

Additionally the IMDG leveraged by the system enabled seamless communication and data exchange among the robots without the need for developing complex and specialized protocols for inter-robot communications. The shared data structures facilitated efficient collaboration and coordination among the robots, enhancing the overall performance and effectiveness of the object transport process.

By leveraging the capabilities of the IMDG, the strategy not only simplifies the communication process but also improves the overall efficiency and reliability of the multi-robot system. This enables robots to effectively share information, adapt to dynamic environments, and execute their tasks collaboratively, ultimately resulting in successful object transport.

In summary, this work introduces a coordinated control strategy for object transport using caging, with the added advantage of using an IMDG for efficient data sharing. The strategy has been demonstrated and validated through simulations, showcasing its effectiveness in achieving successful object transportation within a multi-robot system. The utilization of the IMDG simplifies interrobot communication, enhances collaboration, and contributes to the overall efficiency of the system.

5.1 Future Work

In the future, several opportunities are envisioned to explore and develop various areas of improvement. Firstly, there is an opportunity to incorporate collision avoidance mechanisms into the cooperative transport strategy. When integrated with obstacle avoidance and sensing capabilities, robots can improve their ability to ensure safe and efficient object transportation.

Additionally, building in robust failure recovery strategies provides another avenue to improve the efficiency of transportation. This will involve development of techniques that will allow the robots to adapt, recover, and continue their cooperative task in the event of unforeseen events or failures that occur during the transport process.

Finally, navigating through narrow passages is another area of improvement that can be focused on. To tackle this, we are interested in exploring the concept of deforming the formation of the robot team to enable successful passage through constrained spaces. By dynamically adjusting their arrangement or shape, robots can effectively navigate narrow areas, ensuring successful transport of the object to its destination.

In addition to the cooperative transport problem, we are also intrigued by the

related challenge of placing a set of objects according to user specifications. By extending this research, we aim to investigate methods and algorithms that will allow robots to autonomously position multiple objects in desired configurations. This includes considering constraints, preferences, and specifications provided by the user, enabling the robots to intelligently arrange the objects in a manner that satisfies the given criteria [53, 31, 10].

In general, our future research direction encompasses collision avoidance, failure recovery strategies, traversing narrow passages, and exploring the problem of assembly. By delving into these areas, we strive to advance the capabilities and autonomy of cooperative robotic systems, paving the way for more efficient and adaptable solutions in real-world applications.

Bibliography

[1] R. Alami et al. "Multi-robot cooperation in the MARTHA project". In: *IEEE Robotics Automation Magazine* 5.1 (1998), pp. 36–47. DOI: 10.1109/100.667325.

[2] Neil Alishev et al. "Network Failure Detection and Autonomous Return Algorithms for a Crawler Mobile Robot Navigation". In: *2018 11th International Conference on Developments in eSystems Engineering (DeSE)*. 2018, pp. 169–174. DOI: 10.1109/DeSE.2018.00040.

[3] Indu Arora and Anu Gupta. "Improving performance of cloud based transactional applications using in-memory data grid". In: *International Journal of Computer Applications* 107.13 (2014).

[4] Levent Bayındır. "A review of swarm robotics tasks". In: *Neurocomputing* 172 (2016), pp. 292–321. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2015.05.116. URL: https://www.sciencedirect.com/science/article/pii/S0925231215010486.

[5] Yifan Cai and Simon X Yang. "A survey on multi-robot systems". In: *World Automation Congress 2012*. IEEE. 2012, pp. 1–6.

[6] Y.U. Cao et al. "Cooperative mobile robotics: antecedents and directions". In: *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and*

Systems. Human Robot Interaction and Cooperative Robots. Vol. 1. 1995, 226–234 vol.1. DOI: 10.1109/IROS.1995.525801.

[7] J. Carlson, R.R. Murphy, and A. Nelson. "Follow-up analysis of mobile robot failures". In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*. Vol. 5. 2004, 4987–4994 Vol.5. DOI: 10.1109/ROBOT.2004.1302508.

[8] Seyoung Cheon, Kwanghyun Ryu, and Yonghwan Oh. "Object manipulation using robot arm-hand system". In: *2013 10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE. 2013, pp. 163–166.

[9] Nikolaus Correll et al. "Analysis and Observations From the First Amazon Picking Challenge". In: *IEEE Transactions on Automation Science and Engineering* 15.1 (2018), pp. 172–188. DOI: 10.1109/TASE.2016.2600527.

[10] Akansel Cosgun et al. "Push planning for object placement on cluttered table surfaces". In: *2011 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2011, pp. 4627–4632.

[11] George Coulouris et al. *Distributed Systems: Concepts and Design*. 5th. USA: Addison-Wesley Publishing Company, 2011. ISBN: 0132143011.

[12] Anwesha Das et al. "Performance analysis of a multi-tenant in-memory data grid". In: *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*. IEEE. 2016, pp. 956–959.

[13] Yong Duan and Xiangyou Yu. "Multi-robot system based on cloud platform". In: *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*. 2016, pp. 614–617. DOI: 10.1109/CGNCC.2016.7828856.

[14] Pavel Dzitac and Abdul Md Mazid. "Factors that influence reliable object manipulation". In: *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*. IEEE. 2013, pp. 1468–1473.

[15] Alexei Edelev et al. "Large-scale analysis of energy system vulnerability using in-memory data grid." In: *ICCS-DE*. 2020, pp. 89–98.

[16] Joel M Esposito. "Decentralized cooperative manipulation with a swarm of mobile robots". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2009, pp. 5333–5338.

[17] A. Farinelli, L. Iocchi, and D. Nardi. "Multirobot systems: a classification focused on coordination". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 34.5 (2004), pp. 2015–2028. DOI: 10.1109/TSMCB.2004.832155.

[18] Alessandro Farinelli, Luca Iocchi, and Daniele Nardi. "Multirobot systems: a classification focused on coordination". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 34.5 (2004), pp. 2015–2028.

[19] Hamed Farivarnejad and Spring Berman. "Multirobot Control Strategies for Collective Transport". In: *Annual Review of Control, Robotics, and Autonomous Systems* 5.1 (2022), pp. 205–219. DOI: 10.1146/annurev-control-042920-095844. eprint: https://doi.org/10.1146/annurev-control-042920-095844. URL: https://doi.org/10.1146/annurev-control-042920-095844.

[20] Inc. Gartner. *In-memory data grids (IMDG) software reviews 2023: Gartner Peer insights*. URL: https://www.gartner.com/reviews/market/in-memory-data-grids.

[21] Brian Gerkey. $slam_gmapping - ROSWiki$. URL: https://wiki.ros.org/slam_gmapping.

[22] Yi Guo, Lynne Parker, and Raj Madhavan. “Towards Collaborative Robots for Infrastructure Security Applications”. In: (Mar. 2004).

[23] Ronny Hartanto and Markus Eich. “Reliable, cloud-based communication for multi-robot systems”. In: *2014 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*. 2014, pp. 1–8. DOI: 10.1109/TePRA.2014.6869142.

[24] *Hazelcast Home (https://hazelcast.com/)*. Apr. 2023. URL: https://hazelcast.com/.

[25] Hengjing He et al. “Cloud based real-time multi-robot collision avoidance for swarm robotics”. In: *International Journal of Grid and Distributed Computing* 9.6 (2016), pp. 339–358.

[26] *In-Memory Data Grid: A Complete Overview*. URL: https://hazelcast.com/glossary/in-memory-data-grid/.

[27] Shan Jiang et al. “Programming large-scale multi-robot system with timing constraints”. In: *2016 25th International Conference on Computer Communication and Networks (ICCCN)*. IEEE. 2016, pp. 1–9.

[28] Alaa Khamis, Ahmed Hussein, and Ahmed Elmogy. “Multi-robot task allocation: A review of the state-of-the-art”. In: *Cooperative robots and sensor networks 2015* (2015), pp. 31–51.

[29] Hiroaki Kitano et al. “RoboCup: the Robot World Cup Initiative”. In: *Proceedings of the International Conference on Autonomous Agents* (Apr. 1998). DOI: 10.1145/267658.267738.

[30] Stefan Kohlbrecher. *hector$_m$apping*. URL: https://wiki.ros.org/hector_mapping.

[31] Gilwoo Lee, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. "Hierarchical planning for multi-contact non-prehensile manipulation". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 264–271.

[32] Sebastian Lehrig, Hendrik Eikerling, and Steffen Becker. "Scalability, Elasticity, and Efficiency in Cloud Computing: A Systematic Literature Review of Definitions and Metrics". In: *Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures*. QoSA '15. Montréal, QC, Canada: Association for Computing Machinery, 2015, pp. 83–92. ISBN: 9781450334709. DOI: 10.1145/2737182.2737185. URL: https://doi.org/10.1145/2737182.2737185.

[33] Lixing Liu et al. "Path planning techniques for mobile robots: Review and prospect". In: *Expert Systems with Applications* 227 (2023), p. 120254. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2023.120254. URL: https://www.sciencedirect.com/science/article/pii/S095741742300756X.

[34] Eitan Marder-Eppstein. *move_base*. URL: https://wiki.ros.org/move_base.

[35] Iván Maza et al. "Experimental results in multi-UAV coordination for disaster management and civil security applications". In: *Journal of Intelligent and Robotic Systems: Theory and Applications*. Vol. 61. 1-4. 2011, pp. 563–585. DOI: 10.1007/S10846-010-9497-5.

[36] Luis Merino et al. "A cooperative perception system for multiple UAVs: Application to automatic detection of forest fires". In: *Journal of Field Robotics* 23 (2006).

[37] Prases K Mohanty et al. “Path Planning Techniques for Mobile Robots: A Review”. In: *International Conference on Soft Computing and Pattern Recognition*. Springer. 2021, pp. 657–667.

[38] R.R. Murphy. “Marsupial and shape-shifting robots for urban search and rescue”. In: *IEEE Intelligent Systems and their Applications* 15.2 (2000), pp. 14–19. DOI: 10.1109/5254.850822.

[39] Priyank Narvekar and Andrew Vardy. “A data grid strategy for non-prehensile object transport by a multi-robot system”. In: *Artif. Life Robot.* 28.4 (Oct. 2023), pp. 680–689. ISSN: 1433-5298. DOI: 10.1007/s10015-023-00908-5. URL: https://doi.org/10.1007/s10015-023-00908-5.

[40] Michal Nemrava and Petr Cermák. “Solving the box-pushing problem by master-slave robots cooperation”. In: *Journal of Automation, Mobile Robotics and Intelligent Systems* (2008), pp. 32–37.

[41] *neo_localization|NeobotixROSDocumentation*. en. URL: https://neobotix-docs.de/ros/packages/neo_localization.html.

[42] Noboru Noguchi et al. “Development of a master–slave robot system for farm operations”. In: *Computers and Electronics in Agriculture* 44.1 (2004), pp. 1–19. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2004.01.006. URL: https://www.sciencedirect.com/science/article/pii/S0168169904000316.

[43] Lynne E. Parker. “Multiple Mobile Robot Systems”. In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 921–941. DOI: 10.1007/978-3-540-30301-5_41. URL: https://doi.org/10.1007/978-3-540-30301-5_41.

[44] Zhifeng Qian et al. “Robot learning from human demonstrations with inconsistent contexts”. In: *Robotics and Autonomous Systems* 166 (2023), p. 104466. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2023.104466. URL: https://www.sciencedirect.com/science/article/pii/S0921889023001057.

[45] Elon Rimon and Andrew Blake. “Caging 2D bodies by 1-parameter two-fingered gripping systems”. In: *Proceedings of IEEE International Conference on Robotics and Automation*. Vol. 2. IEEE. 1996, pp. 1458–1464.

[46] *Robot Operating System - ROS, http://wiki.ros.org/*. URL: https://www.ros.org/.

[47] *ROS tf, http://wiki.ros.org/tf*. URL: http://wiki.ros.org/tf.

[48] Haytham Salhi et al. “Open Source In-Memory Data Grid Systems: Benchmarking Hazelcast and Infinispan”. In: *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*. ICPE ’17. L’Aquila, Italy: Association for Computing Machinery, 2017, pp. 163–164. ISBN: 9781450344043. DOI: 10.1145/3030207.3053671. URL: https://doi.org/10.1145/3030207.3053671.

[49] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. “Robotic grasping of novel objects using vision”. In: *The International Journal of Robotics Research* 27.2 (2008), pp. 157–173.

[50] J Spletzer et al. “Cooperative localization and control for multi-robot manipulation”. In: *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*. Vol. 2. IEEE. 2001, pp. 631–636.

[51] Volker Strobel and Marco Dorigo. "Blockchain Technology for Robot Swarms: A Shared Knowledge and Reputation Management System for Collective Estimation". In: *Swarm Intelligence – Proceedings of ANTS 2018 – Eleventh International Conference*. Ed. by Marco Dorigo et al. Vol. 11172. LNCS. Cham, Switzerland: Springer, 2018, pp. 425–426. URL: https://link.springer.com/book/10.1007/978-3-030-00533-7.

[52] A. Stroupe et al. "Behavior-based multi-robot collaboration for autonomous construction tasks". In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005, pp. 1495–1500. DOI: 10.1109/IROS.2005.1545269.

[53] Jochen Stüber, Claudio Zito, and Rustam Stolkin. "Let's Push Things Forward: A Survey on Robot Pushing". In: *Frontiers in Robotics and AI* 7 (2020).

[54] Andrey Tapekhin, Igor Bogomolov, and Oleg Velikanov. "Analysis of Consistency for In Memory Data Grid Apache Ignite". In: *2019 Ivannikov Memorial Workshop (IVMEM)*. 2019, pp. 46–50. DOI: 10.1109/IVMEM.2019.00013.

[55] Elio Tuci, Muhanad HM Alkilabi, and Otar Akanyeti. "Cooperative object transport in multi-robot systems: A review of the state-of-the-art". In: *Frontiers in Robotics and AI* 5 (2018), p. 59.

[56] Israel A. Wagner and Alfred M. Bruckstein. "Cooperative Cleaners: A Study in Ant Robotics". In: *Communications, Computation, Control, and Signal Processing: a tribute to Thomas Kailath*. Ed. by Arogyaswami Paulraj, Vwani Roychowdhury, and Charles D. Schaper. Boston, MA: Springer US, 1997, pp. 289–308. ISBN: 978-1-4615-6281-8. DOI: 10.1007/978-1-4615-6281-8_16. URL: https://doi.org/10.1007/978-1-4615-6281-8_16.

[57] Sonia Waharte, Niki Trigoni, and Simon Julier. "Coordinated Search with a Swarm of UAVs". In: *2009 6th IEEE Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops*. 2009, pp. 1–3. DOI: 10.1109/SAHCNW.2009.5172925.

[58] Yuan Wang et al. "A Scalable Queuing Service Based on an In-Memory Data Grid". In: *2010 IEEE 7th International Conference on E-Business Engineering*. 2010, pp. 236–243. DOI: 10.1109/ICEBE.2010.100.

[59] ZhiDong Wang and Vijay Kumar. "Object closure and manipulation by multiple cooperating mobile robots". In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*. Vol. 1. IEEE. 2002, pp. 394–399.

[60] Wikipedia contributors. *Inflection point — Wikipedia, The Free Encyclopedia*. [Online; accessed 16-October-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Inflection_point&oldid=1166315551.

[61] Alan FT Winfield. "Foraging robots". In: (2009).

[62] A. Yamashita et al. "Motion planning of multiple mobile robots for Cooperative manipulation and transportation". eng. In: *IEEE transactions on robotics and automation* 19.2 (2003), pp. 223–237. ISSN: 1042-296X.

[63] Atsushi Yamashita et al. "Motion planning of multiple mobile robots for cooperative manipulation and transportation". In: *IEEE Transactions on Robotics and Automation* 19.2 (2003), pp. 223–237.

[64] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. "A Survey and Analysis of Multi-Robot Coordination". In: *International Journal of Advanced Robotic Systems* 10.12 (2013), p. 399. DOI: 10.5772/57313. eprint: https://doi.org/10.5772/57313. URL: https://doi.org/10.5772/57313.

[65] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. “A survey and analysis of multi-robot coordination”. In: *International Journal of Advanced Robotic Systems* 10.12 (2013), p. 399.

[66] Chika O. Yinka-banjo and Antoine B. Bagula. “Autonomous multi-robot behaviours for safety inspection under the constraints of underground mine terrains”. In: 2012.

[67] Kai Zhu and Tao Zhang. “Deep reinforcement learning based mobile robot navigation: A review”. In: *Tsinghua Science and Technology* 26.5 (2021), pp. 674–691. DOI: 10.26599/TST.2021.9010012.

[68] DA Zlobin. “In-Memory Data Grid”. In: (2017).