Degree Project in Computer Science and Engineering, specializing in Machine
Learning
Second cycle, 30 credits

Multi-robot coordination and planning with human-in-the-loop under STL specifications
Centralized and distributed frameworks

YIXIAO ZHANG
Multi-robot coordination and planning with human-in-the-loop under STL specifications

Centralized and distributed frameworks

YIXIAO ZHANG

Date: June 22, 2023

Supervisor: Victor Nan Fernandez-Ayala
Examiner: Dimos V. Dimarogonas
   School of Electrical Engineering and Computer Science
Swedish title: Multi-robotkoordination och planering med människlig interaktion under STL-specifikationer
Swedish subtitle: Centraliserade och distribuerade ramverk
Abstract

Recent urbanization and industrialization have brought tremendous pressure and challenges to modern autonomous systems. When considering multiple complex tasks, cooperation and coordination between multiple agents can improve efficiency in a system. In real-world applications, multi-agent systems (MAS) are widely used in various fields, such as robotics, unmanned aerial systems, autonomous vehicles, distributed sensor networks, etc. Unlike traditional MAS systems based on pre-defined algorithms and rules, a special human-in-loop (HIL) based MAS involves human interactions to enhance the system’s adaptability for special scenarios, as well as apply human preferences for robot control.

However, existing HIL strategies are primarily based on human involvement at a low level, such as mixed-initiative control and mixed-agent scenarios with both human-driven and intelligent robots. There are fewer investigations on applying HIL in high-level coordination. In particular, designing a coordination strategy for multi-task multi-agent scenarios, which can also deal with real-time human commands, will be one of the key topics of this Master’s thesis project.

In this thesis work, different kinds of tasks described by signal temporal logic (STL) are created for agents, which can be enforced by control barrier function (CBF) constraints. Both centralized and distributed frameworks are designed for agent coordination. In detail, the centralized strategy is developed for machine-to-infrastructure (M2I) communication, by using the nonlinear model predictive control (NMPC) method to obtain collision-free trajectories. The distributed strategy utilizing graph theory is proposed for machine-to-machine (M2M), in order to reduce computation time by offloading. Most importantly, a HIL model is generated for both frameworks to apply online human commands to the coordination, with a novel task allocation protocol.

Simulations and experiments are carried out on both Matlab and Python-based ROS simulators, to show that proposed frameworks can achieve obvious performance advantages in safety, smoothness, and stability for task completion. Numerical results are provided to validate the feasibility and applicability of our algorithms.

Keywords

Multi-agent systems, Human-in-the-loop systems, Signal temporal logic, Cooperative control, ROS Implementation.
ii | Abstract
**Sammanfattning**

Den senaste urbaniseringen och industrialiseringen har medfört enormt tryck och utmaningar för moderna autonoma system. Vid beaktande av flera komplexa uppgifter kan samarbete och samordning mellan flera agenter förbättra effektiviteten i ett system. I verkliga tillämpningar används multiagent-system (MAS) i stor utsträckning inom olika områden, såsom robotik, obemannade luftfarkoster, autonoma fordon, distribuerade sensorsystem etc. Till skillnad från traditionella MAS-system baserade på fördefinierade algoritmer och regler, innebär ett särskilt människa-i-loop (HIL)-baserat MAS mänsklig interaktion för att förbättra systemets anpassningsförmåga till speciella scenarier samt anpassa mänskliga preferenser för robotstyrning.


Simuleringar och experiment utförs på både Matlab och Python-baserade ROS-simulatorer för att visa att de föreslagna ramverken kan uppnå tydliga prestandafördelar när det gäller säkerhet, smidighet och stabilitet för uppgiftsslutförande. Numeriska resultat presenteras för att validera genomförbarheten och tillämpligheten hos våra algoritmer.
Nyckelord
Multi-agent-system, Människa-i-loop-system, Signaltemporallogik, Samarbetande styrning, ROS-implementering
Acknowledgments

I would like to thank my supervisor, Victor Nan Fernandez-Ayala, for the guidance on both the theoretical and experimental aspects of the thesis. I also would like to thank my examiner, Dimos V. Dimarogonas for reviewing and giving suggestions for my work.

Stockholm, June 2023
Yixiao Zhang
# Contents

1 **Introduction**  
   1.1 Background .................................................. 2  
   1.2 Problem ......................................................... 2  
      1.2.1 Problem Definition ........................................ 3  
      1.2.2 Scientific and engineering issues ...................... 3  
   1.3 Purpose ......................................................... 4  
   1.4 Goals .......................................................... 4  
   1.5 Research Methodology ......................................... 5  
   1.6 Delimitations .................................................. 5  
   1.7 Structure of the thesis ....................................... 6  

2 **Background**  
   2.1 Multi-agent Systems ........................................... 7  
      2.1.1 Centralized Coordination Framework ..................... 8  
      2.1.2 Distributed Coordination Framework ..................... 9  
   2.2 Robot Tasks .................................................. 10  
      2.2.1 Signal Temporal Logic (STL) ............................. 11  
      2.2.2 Splitting and Merging of STL Tasks .................... 12  
   2.3 Control Barrier Functions (CBFs) ............................ 12  
      2.3.1 Definitions for CBFs .................................... 13  
      2.3.2 Create CBFs for STL tasks ............................... 13  
   2.4 Human-in-loop (HIL) Control ................................. 15  
   2.5 Summary ....................................................... 16  

3 **Methods**  
   3.1 Research Process ............................................... 19  
   3.2 Data Collection, Analysis and Measurements .................. 20  
   3.3 Experimental Design ........................................... 21  
      3.3.1 Test Environments ........................................ 21
3.3.2 Simulators to be used ........................................ 22
  3.3.2.1 Matlab Simulator ................................... 22
  3.3.2.2 CANOPIES ROS simulator ......................... 23
3.4 Assessing reliability and validity of the data collected ...... 24
  3.4.1 Reliability and Validity of method ...................... 24
  3.4.2 Reliability and Validity of data ....................... 25

4 MAS Coordination System based on STL tasks ................. 27
  4.1 Problem Formulation ....................................... 27
  4.2 CBF generator for Different Tasks ......................... 28
    4.2.1 Collision Avoidance for obstacles .................. 28
    4.2.2 Collision Avoidance between Agents ................ 29
    4.2.3 Reaching Targets .................................... 30
    4.2.4 CBFs for Other Tasks ............................... 31
  4.3 Coordination Frameworks .................................... 32
    4.3.1 Centralized Coordination Framework .................. 32
    4.3.2 Distributed Coordination Framework .................. 35
  4.4 Human-in-loop (HIL) for Coordination ..................... 37
    4.4.1 HIL Model ........................................... 38
    4.4.2 Task Allocation in Coordination ...................... 39

5 Results and Analysis ........................................... 43
  5.1 Major results .............................................. 43
    5.1.1 Robot Coordination .................................. 43
    5.1.2 HIL tasks ............................................ 45
    5.1.3 ROS-based Experiments ............................... 46
  5.2 Validity and Reliability Analysis .......................... 48

6 Conclusions and Future work .................................. 51
  6.1 Conclusions ................................................ 51
  6.2 Limitations ................................................ 52
  6.3 Future work ................................................. 52
    6.3.1 What has been left undone? ........................... 52
      6.3.1.1 Consensus in Distributed framework .............. 53
      6.3.1.2 Double spending problem in distributed  
                 framework ....................................... 53
      6.3.1.3 Queue stability of tasks ........................ 53
    6.3.2 Next obvious things to be done ....................... 54
  6.4 Reflections ................................................ 55
References

A  Solver for NMPC  61
   A.1  Linear Dynamics  61
   A.2  Nonlinear Dynamics  63
List of Figures

1.1 Coordination Framework and HIL system to be developed. 3

2.1 Graphic models of coordination systems. a) Centralized network for centralized coordination framework. b) Ad Hoc Network for distributed coordination framework. 9

2.2 Graphic Model of CBF. 13

3.1 Diagram of Research process. 20

3.2 Example of Matlab simulator. 23

3.3 Robot model of CANOPIES ROS simulator. 24

4.1 Safety region for static obstacle avoidance. 29

4.2 Time corridor for collision avoidance between agents. 30

4.3 CBF change on time for reaching the target. 31

4.4 Structure of the centralized coordination framework. 32

4.5 Structure of the distributed coordination framework. 35

4.6 Undirected graph for bidirectional communication. 35

5.1 Performance Comparison. 45

5.2 Results on HIL command "All agents follow me later". 46

5.3 Results on HIL command "One agent go to [10, 20]^T now". 47

5.4 Scenarios in the CANOPIES ROS Simulator. 48

5.5 Coordination results for centralized framework. 49

5.6 Coordination results for distributed framework. 50

6.1 Structure of double-level coordination framework. 54
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>STL Semantics [19]</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Comparisons</td>
<td>17</td>
</tr>
<tr>
<td>3.1</td>
<td>Experiments for coordination frameworks</td>
<td>22</td>
</tr>
<tr>
<td>3.2</td>
<td>Experiments for HIL system.</td>
<td>22</td>
</tr>
<tr>
<td>4.1</td>
<td>Dictionary of human task command (TC).</td>
<td>39</td>
</tr>
<tr>
<td>4.2</td>
<td>Adjust the soft task lists based on task priority command (PC).</td>
<td>39</td>
</tr>
<tr>
<td>4.3</td>
<td>Scheduling processes for different subject commands (SCs).</td>
<td>40</td>
</tr>
<tr>
<td>5.1</td>
<td>Simulation tasks for coordination frameworks.</td>
<td>44</td>
</tr>
<tr>
<td>5.2</td>
<td>Calibration of main parameters.</td>
<td>44</td>
</tr>
</tbody>
</table>
List of Tables
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBF</td>
<td>Control Barrier Function</td>
</tr>
<tr>
<td>CN</td>
<td>Centralized Node</td>
</tr>
<tr>
<td>HIL</td>
<td>Human-in-loop</td>
</tr>
<tr>
<td>M2I</td>
<td>Machine-to-Infrastructure</td>
</tr>
<tr>
<td>M2M</td>
<td>Machine-to-Machine</td>
</tr>
<tr>
<td>MAS</td>
<td>Multi-agent Systems</td>
</tr>
<tr>
<td>MEC</td>
<td>Multi-access Edge Computing</td>
</tr>
<tr>
<td>MPC</td>
<td>Model Predictive Control</td>
</tr>
<tr>
<td>NMPC</td>
<td>Nonlinear Model Predictive Control</td>
</tr>
<tr>
<td>PC</td>
<td>Task Priority Command</td>
</tr>
<tr>
<td>ROS</td>
<td>Robot Operating System</td>
</tr>
<tr>
<td>SC</td>
<td>Subject Command</td>
</tr>
<tr>
<td>STL</td>
<td>Signal Temporal Logic</td>
</tr>
<tr>
<td>TC</td>
<td>Task Command</td>
</tr>
</tbody>
</table>
List of Symbols Used

The following symbols will be later used within the body of the thesis.

\( \delta \)  
Priority symbol of the agent, see equation (4.17), page 33

\( \Delta t \)  
Control interval, page 28

\( \Delta t_p \)  
Planning interval, page 34

\( \epsilon \)  
Relaxation factor of the CBF constraint, see equation (4.18), page 33

\( \hat{X} \)  
Previous trajectory of the agent, page 32

\( \hat{h}, \hat{b} \)  
CBF and time-based CBF for soft tasks, page 30

\( p \)  
Agent position in Cartesian Coordinates, page 29

\( \mathcal{L} \)  
Ranking list for saving the agent orders, see equation (4.17), page 33

\( \mathcal{N}^i \)  
Neighbors of agent i in the undirected graph, page 35

\( \mathcal{N}_h \)  
Agent list for human task, page 38

\( b(x, t) \)  
Time-based CBF of a STL task, see equation (2.12), page 14

\( b_{\text{all}}(x, t) \)  
Total CBF of multiple STL tasks, see equation (2.15), page 15

\( \phi_1 U_{[a,b]} \phi_2 \)  
"Until" task in STL, see equation (2.1), page 11

\( \phi, \hat{\phi}, \tilde{\phi} \)  
Agent task, agent soft task, agent hard task, page 27

\( \psi, \hat{\psi}, \tilde{\psi} \)  
Human task, human soft task, human hard task, page 38

\( \hat{U} \)  
Planned Controls in all predicted steps, see equation (4.4), page 28

\( \hat{u} \)  
Planned Control, see equation (4.4), page 28
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bar{X}$</td>
<td>Planned states in all predicted steps, see equation (4.5)</td>
<td>28</td>
</tr>
<tr>
<td>$\bar{x}$</td>
<td>Planned state, see equation (4.5)</td>
<td>28</td>
</tr>
<tr>
<td>$\tilde{h}, \tilde{b}$</td>
<td>CBF and time-based CBF for hard tasks</td>
<td>29</td>
</tr>
<tr>
<td>$f, g$</td>
<td>Functions in the input-affine nonlinear control system</td>
<td>27</td>
</tr>
<tr>
<td>$F_{[a,b]}$</td>
<td>”Eventually” task in STL, see equation (2.1)</td>
<td>11</td>
</tr>
<tr>
<td>$G_{[a,b]}$</td>
<td>”Always” task in STL, see equation (2.1)</td>
<td>11</td>
</tr>
<tr>
<td>$h(x)$</td>
<td>Classical CBF, see equation (2.5)</td>
<td>13</td>
</tr>
<tr>
<td>$M$</td>
<td>Ranking function for agent priorities, see equation (4.16)</td>
<td>32</td>
</tr>
<tr>
<td>$N$</td>
<td>The number of agents</td>
<td>27</td>
</tr>
<tr>
<td>$n_h$</td>
<td>Number of agents for human task</td>
<td>38</td>
</tr>
<tr>
<td>$N_p$</td>
<td>Length of the predicted horizon</td>
<td>28</td>
</tr>
<tr>
<td>$R_{reg}$</td>
<td>Radius for the target region, see equation (4.12)</td>
<td>30</td>
</tr>
<tr>
<td>$R_{ob}$</td>
<td>Radius of the circular obstacle, see equation (4.6)</td>
<td>29</td>
</tr>
<tr>
<td>$R_{red}$</td>
<td>Radius for redundancy</td>
<td>29</td>
</tr>
<tr>
<td>$R_{safe}$</td>
<td>Safe distance</td>
<td>29</td>
</tr>
<tr>
<td>$t_c$</td>
<td>Current time</td>
<td>28</td>
</tr>
<tr>
<td>$u_{max}$</td>
<td>Speed limitation, see equation (4.15)</td>
<td>31</td>
</tr>
<tr>
<td>$x, u$</td>
<td>The agent state and control input</td>
<td>27</td>
</tr>
<tr>
<td>$\Phi_H$</td>
<td>The task list of agent hard tasks, see equation (4.2)</td>
<td>27</td>
</tr>
<tr>
<td>$\Phi_S$</td>
<td>The task list of agent soft tasks, see equation (4.1)</td>
<td>27</td>
</tr>
<tr>
<td>$\Psi_H$</td>
<td>The task list of human hard tasks, see equation (4.30)</td>
<td>38</td>
</tr>
<tr>
<td>$\Psi_H$</td>
<td>The task list of human soft tasks, see equation (4.29)</td>
<td>38</td>
</tr>
<tr>
<td>$p_t$</td>
<td>Center position of the target region, see equation (4.12)</td>
<td>30</td>
</tr>
<tr>
<td>$p_{ob}$</td>
<td>Center position of the circular obstacle, see equation (4.6)</td>
<td>29</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In this thesis work, we will focus on multi-robot coordination based on Signal Temporal Logic (STL) for safe multi-agent navigation, as well as Human-in-loop (HIL) control for human preference. To be specific, the contributions can be categorized into two parts. The first part is for multi-agent coordination. STL is used to specify complex robot objectives, such as periodic, sequential, or reactive tasks. Control Barrier Functions (CBFs) can present decentralized functions for safe multi-robot constraints. Additionally, both centralized and distributed coordination strategies are under our consideration. For the centralized one, a Multi-access Edge Computing (MEC) server is used to share information through Machine-to-Infrastructure (M2I) communication, as well as compute feasible trajectories for different agents. Furthermore, Machine-to-Machine (M2M) communication based on graph theory can provide a real-time communication network for distributed planning systems. The second part is for HIL control. This would be implemented mainly through online re-planning based on Nonlinear Model Predictive Control (NMPC), where the human will create real-time tasks (also called human predicates) and the robots will have to adapt their previous offline plan to also achieve these new human predicates. Collision avoidance between agents and humans will also need to be considered in the system.

The algorithm validations and performance comparisons will be carried out based on the proposed frameworks. Furthermore, the representative vineyard scenarios from the EU CANOPIES 3D Simulator by PaleBlue [1] will be used in order to further verify the feasibility and practicability of the proposed methods in real-world applications.
1.1 Background

In recent years, with the continuous development of intelligent systems, multi-agent cooperation research has entered a new stage and become the cornerstone of modern society. People’s demand for robots to complete complex tasks is gradually increasing, but a single agent cannot complete some tasks such as joint handling and planning, which need to be completed by multiple agents. However, at intersections where multiple routes cross or merge, collisions and deadlocks between robots are especially pronounced, which reduces safety and efficiency [2]. Therefore, robot coordination is one of the most pressing and challenging problems in Multi-agent Systems (MAS).

Furthermore, considering that humans participate in the system as supervisors or collaborators, the co-adaptation between robots and humans also needs to be carried out. HIL planning can help to accomplish tasks more efficiently and safely [3]. At a high level, the human operators can assign tasks for robot coordination, or monitor the progress of the tasks at run time. At a low level, humans can directly influence the control orders and even help the low-level planner adapt to some specific driving patterns by imitation learning.

1.2 Problem

For multi-agent management, efficiency and safety are important indexes to measure the performance and feasibility of the systems. (i) Efficiency refers to the ability of an agent to complete tasks. By reasonable task allocation to agents and effective coordination of their actions, they can perform all the tasks within the specified time requirements. (ii) Safety refers to the collision avoidance between agents and obstacles. Multi-agents not only need to consider the static and dynamic obstacles that may exist in the environment but also need to prevent trajectory overlaps between agents.

A coordination strategy is very important for MAS to achieve both efficiency and safety. But its design strongly depends on the way information interacts between agents. According to different communication patterns and restrictions, our coordination framework needs to be adapted and adjusted.

Another important scenario is HIL. In practice, many tasks are not assigned to agents in advance. Humans may provide high-level real-time verbal commands based on the actual situations. Thus, our framework needs to be able to process new human instructions dynamically, which demonstrates the flexibility required by the problem for the system.
1.2.1 Problem Definition

Consider there are $N$ agents who participated in the scenario and each agent has some initial STL tasks. The problem is designing coordination frameworks based on communication types to achieve feasible control for agents. CBF constraints can be used in the optimization to fulfill different STL requirements without any collisions and deadlocks. Another is the HIL problem we need to consider for resolving online human commands, where task allocation between agents is needed. The following diagram in Fig. 1.1 shows the overall structure of our system.

![Coordination Framework and HIL system to be developed.](image)

1.2.2 Scientific and engineering issues

In order to apply the system to engineering scenarios, an important problem is to describe the robot and human tasks as STL formulas that can be processed by the algorithms. By generating CBF constraints for STL formulas, we can limit the agents in their safe and feasible areas in coordination from the mathematical point of view.

Considering the communication delay and jitter in practical engineering, as well as the computing power limitation, we need to choose between a centralized or distributed strategy. The centralized framework is usually deployed at base stations or in the cloud, which has a powerful computing server but requires high real-time communication. The distributed framework can offload the computation to each agent. Although the computing power is reduced, a more flexible method of network access can be used.
In addition, according to the time complexity of the algorithms and the requirements of control error, we can reasonably choose the prediction horizon, control, and planning interval of Model Predictive Control (MPC) in the optimization, for the usages in different environments.

1.3 Purpose

The purpose of coordination in a MAS is to enable agents to work together in a cooperative and efficient manner to achieve a common goal or task. This kind of coordination promotes resource sharing among agents, and at the same time assigns specific tasks according to their abilities, which can improve the fairness and efficiency of resource utilization and management. Furthermore, it can resolve conflicts that may arise due to differences in goals or priorities between agents, and improve overall robustness.

However, even if MAS has all the advantages mentioned above, MAS is difficult to deal with complex or ambiguous objectives and unexpected situations that may arise in the system. Humans can monitor the behavior of MAS, detect anomalies, identify potential risks based on visual feedback, and then provide high-level commands to agents in human critical situations through expertise and decision-making ability. Therefore, adding HIL to MAS can improve system performance and adaptability.

Through literature investigation in related fields, this paper aims to design coordination frameworks and add the HIL model to provide a robust solution in the multi-task multi-agent environment while ensuring security.

1.4 Goals

The goal of this project is to create both centralized and distributed coordination frameworks for STL tasks provided by task allocators and human commands. This has been divided into the following three sub-goals:

1. Subgoal 1: Get familiar with STL, CBF, MPC, and NMPC for ego robot planning and control;

2. Subgoal 2: Investigate several multi-robot coordination algorithms, as well as their solvers and offloading strategies;

3. Subgoal 3: Design both centralized and distributed coordination frameworks for STL tasks based on CBF constraints, and then validate them theoretically.
4. Subgoal 4: Apply HIL into the previous frameworks.

5. Subgoal 5: Implement the proposed algorithms on multi-robot multi-task scenarios in the Matlab simulator to verify their feasibility;

6. Subgoal 6: Get familiar with the CANOPIES ROS simulator and apply the multi-robot coordination algorithms in the agri-food area to show their practicability.

1.5 Research Methodology

The main methods to be used throughout the thesis are system and mathematical model designs. Firstly, task-based coordination frameworks adopt the CBF-based control scheme, which has been proven to be stable. In distributed strategy, we provide a priority allocation strategy based on graph theory. Furthermore, a model for human instructions is provided based on the logical description using STL. Secondly, Matlab implementations will be used to test experimentally and safely. Python implementations on the ROS-based simulator show that frameworks are applicable to real-world scenes.

1.6 Delimitations

Due to time limitations on the thesis, our distributed coordination framework assumes that the communication is ideal within a certain circular range, and we only consider one-hop communication in the ad hoc network. For the centralized strategy, we assume that all agents can ideally connect to the MEC. We do not specifically study the impact of communication protocols, delay, and packet loss rate on the planning results.

In addition, this study mainly focuses on movement-based tasks, such as going to a point, and following a moving object. We do not consider other types of tasks, such as robotic arm grasping and placing.

The human commands in this paper consider only one of the most common command structures "somebody to do something at some time”. Other types of commands with more complex logic combined with speech and gesture recognition will be researched in future work.
1.7 Structure of the thesis

Chapter 2 presents relevant background information about MAS coordination, temporal logics, CBFs, and HIL approaches. Chapter 3 presents the methodology and method used to solve the problem. Chapter 4 introduces the principles of the main centralized and distributed frameworks in this thesis. Chapter 5 presents and analyzes the numerical results of simulations and experiments. Chapter 6 concludes the thesis and gives some extra information about future work.
Chapter 2

Background

This chapter provides basic background information about planning and coordination in MAS. Additionally, this chapter also describes some basic ideas about STL, CBFs as well as some related works on HIL tasks.

2.1 Multi-agent Systems

In the field of robotics, there is plenty of research on single-agent control. In these studies, an ego agent only collects environmental information through its onboard sensors for decision-making and planning. However, when we consider some real-world applications with more complex objectives, MAS is facing tremendous attention with rapid development. Different tasks are allocated to a number of autonomous agents, and proper cooperation among them is helpful in achieving goals smoothly, efficiently, and safely. Each agent makes decisions and plans not only according to its own state, targets, and observations but also based on information interactions between agents, in order to obtain the current or future actions [4].

There are many applications for using MAS in real-world scenarios. One of the most common examples is in the field of industrial machines, such as factory packaging. Multiple robots will pick up and distribute packages according to their location and destination. Efficient allocation and routing strategies can save labor and time costs [5]. MAS is also popular in autonomous driving. When we consider multiple intelligent vehicles entering road intersections, a flexible coordination framework can help to avoid traffic accidents as well as jams [6]. Furthermore, MAS can also be applied to UAV systems, sensor networks, social networks, and other fields to complete different tasks such as formation, data analysis, and behavior prediction.
Efficiency and security are important indicators of the MAS system. Efficiency means that, through the interaction of different individuals, robots can increase the number of tasks completed per unit of time. For example, in a traffic environment, traffic throughput can measure road congestion. Through reasonable coordination, we can increase the number of vehicles passing through per unit of time, which can improve traffic efficiency. At the same time, while improving efficiency, ensuring that no collision occurs or maintaining a certain safe distance between agents can ensure the feasibility of the MAS. One common way is defining a set of mathematical rules and constraints that determine how robots should behave to prioritize safety, for example, through CBF which is a useful tool to describe distance constraints between agents.

According to different communication modes, the coordination problem of MAS can be divided into two types, centralized and distributed frameworks, which we will analyze in detail.

### 2.1.1 Centralized Coordination Framework

For centralized approaches, we normally believe that there is a Centralized Node (CN) in the environment, which can receive requests from agents and then coordinate the maneuver and trajectory for each agent through M2I communication. In reality, CN could be a cloud or a base station with a MEC server. The connection between agents and CN can be either a direct connection or using edge node base stations (eNBs) deployed in the road site unit (RSU) as relays, as shown in Fig. 2.1 a).

We can define \( N \) as the total number of agents accessing MEC. At CN, with the information collected by all agents \( \{ I_j(t) \}_{j \in \{1, 2, ..., N\}} \), such as states, tasks, and observations, the high-level planner on MEC can plan the optimal control orders for each agent to avoid collision as well as achieve its local goals.

Centralized coordination has many applications in vehicular networks based on 802.11p communication [7]. To guarantee agent safety, one common approach is to discretize the space and define the critical sets, which are areas for possible collisions. Mixed integer programming (MIP) can be used in CN to ensure that only one agent is allowed to occupy a set simultaneously [8]. Another approach is to turn the coordination into a resource allocation problem. CN can allocate different time and space resources to each agent to avoid trajectory conflicts and deadlocks [9]. Centralized control is also applied in robotic tasks such as formation and tracking, by combining multi-agent states and designing collaborative feedback control law [10] [11].
The advantage of centralized coordination is that it can synthesize all agent information and obtain solutions closer to the global optimal. At the same time, the algorithms can avoid local deadlocks and provide a more robust security guarantee. Furthermore, centralized approaches are more simple to implement and deploy with some existing facilities, such as base stations and cell towers. On the other hand, the centralized frameworks also have numerous drawbacks. Firstly, the communication, computation, and storage resources required by the centralized coordinator increase with the number of agents and the scale of the network. Thus, the scalability of the methods is poor. Secondly, the methods depend on the functionality of the infrastructure. When the base station fails, it will cause a large number of agents to lose control. In addition, the communication delay is also a big factor in the feasibility, and information security on public networks will prevent these methods from being applied to sensitive tasks.

![Figure 2.1: Graphic models of coordination systems. a) Centralized network for centralized coordination framework. b) Ad Hoc Network for distributed coordination framework.](image)

**2.1.2 Distributed Coordination Framework**

Distributed approaches are based on the ad hoc network, which is a collection of moving nodes. Each agent in the system can be viewed as a mobile node, which can join the network or leave the network at any time. Each node in the network can act as either a communication host or a router, relaying data packets to other hosts, as shown in Fig. 2.1 b). According to the specific communication protocol and application scenario, we can decide to use one-hop or multi-hop transmission in the network.
Graph theory is a useful tool to describe the communication graph in MAS [2]. A digraph can describe as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, ..., N\}$ is the agent set and $\mathcal{E} = \mathcal{V} \times \mathcal{V}$ is the edge set. When we consider only one-hop communication, an edge $(i, j) \in \mathcal{E}$ means that $j$ can send information to $i$, where $i$ is regarded as an in-neighbor of $j$ and $j$ is regarded as an out-neighbor of $i$. If $\forall (i, j) \in \mathcal{E}, (j, i) \in \mathcal{E}$, the graph is defined as an undirected graph. Therefore, for each agent $i, i = 1, 2, ..., N$, it can obtain its own state $x_i(t)$ and information passed by its out-neighbors $N_{\text{out}}i$, which can be represented as $\{I_j(t)\}_{j \in N_{\text{out}}i}$, to plan a feasible trajectory on onboard computing unit for real-time control.

There also exist several investigations on distributed planning algorithms. Optimization-based coordination is the most popular approach, which can transform tasks and limitations into goals and constraints, then obtain the optimal solution in a distributed way through the exchange of local information and iterative calculation [12] [13]. Consensus is achieved when the participating entities converge to a shared decision or state, which is widely used in the distributed formation control [14]. Auction-based coordination is another kind of popular algorithm. Each agent can participate in the auction according to its own ability and resources, and decide whether to accept tasks or provide resources according to the auction results [15]. In addition, for some tasks based on agent-based confrontation, such as pursuit and evasion, we can use game theory to choose the optimal strategy. Reinforcement learning-based algorithms are also widely used in gaming tasks [16].

2.2 Robot Tasks

For diverse robot tasks, we need a common language to specify the objective or desired outcome of the agents. Considering the time and order requirements between tasks, the language also needs to specify time constraints on the execution of tasks. Therefore, temporal logic is widely used to define complex tasks for different agents. Existing approaches include linear temporal logic (LTL) and signal temporal logic (STL). LTL mainly describes the properties of discrete time series, which can define the relationship between future and current states in the sequence of events [17]. STL is an extended temporal logic that offers robust semantics which can not only describe the properties of discrete time series but also deal with the properties of continuous-time signals. STL also introduces signal comparison and temporal operators, which can describe signals’ values and temporal attributes [18]. In this paper, we focus on multi-agent coordination for STL tasks.
### 2.2.1 Signal Temporal Logic (STL)

In STL [18], we can evaluate a continuously differentiable predicate function \( h : \mathbb{R}^n \rightarrow \mathbb{R} \) to decide predicate \( \mu \), which can be described as

\[
\mu = \begin{cases} 
\text{true}, & h(x) \geq 0 \\
\text{false}, & h(x) < 0 
\end{cases}
\]  

(2.1)

In this paper, we can define an STL formula based on the STL syntax, i.e.,

\[
\phi := \text{true} \mid \neg \phi \mid \phi_1 \land \phi_2 \mid G_{[a,b]} \phi \mid F_{[a,b]} \phi \mid \phi_1 U_{[a,b]} \phi_2 
\]  

(2.2)

where \( \phi, \phi_1 \) and \( \phi_2 \) are STL formulas. \( 0 \leq a \leq b \) is the time interval of the different STL operators including always \( G \), eventually \( F \), and until \( U \). We denote \( (x, t) \models \phi \) as the signal \( x : \mathbb{R}^n \rightarrow \mathbb{R} \) satisfying \( \phi \) at time \( t \).

Thus, the STL semantics can be defined as Table 2.1. It is worth noting that in our scenarios, we do not include the "Disjunction" \( \phi_1 \lor \phi_2 \) operator (do task 1 or task 2) in the list. That is because of the difficulty in actually representing it in a simple way as a predicate function. For the "Conjunction" operator, we can represent it by finding the smallest CBF value, which will be described in detail later. But for the "Disjunction" operator, we need to judge and choose between two CBFs, which is difficult to describe the overall CBF by a single function with the consideration of different choosing criteria. However, in our cases, we can include the "or" operation through high-level task allocation between agents. Thus, the comparison can be executed in the coordinator, without increasing the complexity of CBFs.

**Table 2.1: STL Semantics [19]**

<table>
<thead>
<tr>
<th>Common</th>
<th>( (x, t) \models \phi )</th>
<th>( h(x(t)) \geq 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negation</td>
<td>( (x, t) \models \neg \phi )</td>
<td>( \neg((x, t) \models \phi) )</td>
</tr>
<tr>
<td>Conjunction</td>
<td>( (x, t) \models \phi_1 \land \phi_2 )</td>
<td>( ((x, t) \models \phi_1) \land ((x, t) \models \phi_2) )</td>
</tr>
<tr>
<td>Always</td>
<td>( (x, t) \models G_{[a,b]} \phi )</td>
<td>( \forall t_1 \in [t + a, t + b], (x, t_1) \models \phi )</td>
</tr>
<tr>
<td>Eventually</td>
<td>( (x, t) \models F_{[a,b]} \phi )</td>
<td>( \exists t_1 \in [t + a, t + b], \text{s.t.} (x, t_1) \models \phi )</td>
</tr>
<tr>
<td>Until</td>
<td>( (x, t) \models \phi_1 U_{[a,b]} \phi_2 )</td>
<td>( \exists t_1 \in [t + a, t + b] \text{s.t.} (x, t_1) \models \phi_2 \land \forall t_2 \in [t, t_1], (x, t_2) \models \phi_1 )</td>
</tr>
</tbody>
</table>

Therefore, we can use the combinations of the STL formulas to describe different robot tasks.
2.2.2 Splitting and Merging of STL Tasks

According to [20] [21], we know that the satisfaction of \( G_{[a,b]} \phi_1 \land F_{[b,b]} \phi_2 \) implies the satisfaction of the "until" task \( \phi_1 U_{[a,b]} \phi_2 \) in equation (2.2). Therefore, when we consider an STL formula \( \phi'' \) with a combination of \( G, F, \) or \( U \) tasks, it can be written as

\[
\phi'_{k} := G_{[a_k,b_k]} \phi | F_{[a_k,b_k]} \phi | \phi_1 U_{[a_k,b_k]} \phi_2 \\
\hat{\phi}'_{k} := G_{[a_k,b_k]} \phi | F_{[a_k,b_k]} \phi \\
\phi'' := \bigwedge_{k=1}^{n_{G}+n_{F}+n_{U}} \phi'_{k} := \bigwedge_{k_1=1}^{n_{G}+n_{U}} \bigwedge_{k_2=1}^{n_{F}+n_{U}} \bigwedge_{k=1}^{n_{G}+n_{F}+2n_{U}} \hat{\phi}'_{k} 
\]

(2.3)  (2.4)  (2.5)

where \( n_{G}, n_{F}, \) and \( n_{U} \) are the number of "always", "eventually", and "until" tasks respectively. \( \phi, \phi_1 \) and \( \phi_2 \) are the simple basic STL formula with the form \( \phi := \text{true} | \mu | \neg \mu \). From the equation (2.5), we know that \( \phi \) can be written as a conjunction of always and eventually tasks.

2.3 Control Barrier Functions (CBFs)

CBF is a powerful tool that provides security constraints for dynamic systems. It can be used to design control laws to ensure that systems operate in safe areas of the state space. CBF defines a function to describe the behavior of the system and establishes a lower bound for the security constraints. The stability of the CBF system can be analyzed using a technique similar to Lyapunov’s.

There are some investigations on CBF, which are applied in a variety of fields, including robotics, autonomous systems, and safety-critical controls to ensure the safety and reliability of complex systems. Authors in [22] provided an overview of the theory and applications of CBFs, including theoretical foundations, stability analysis, and practical implementation in robotic systems. Authors in [23] used CBFs to assist the UAV to avoid obstacles, thus achieving safe remote control flight. In [24], discrete-time CBF constraints were created to avoid obstacles between polytopes, which can be used in the autonomous vehicle field.

In this paper, we focus on creating suitable CBFs for STL tasks and collision avoidance in MAS, which is useful to achieve efficiency and safety for our coordination systems.
2.3.1 Definitions for CBFs

The essence of CBF is to define a function that specifies boundaries or conditions that the system should not violate. As shown in Fig. 2.2, when an agent has a bounded safe region, we can define a differentiable function $h(x)$ that has a value greater than 0 in the interior of the safe region, a value equal to 0 on the boundary, and a value less than 0 in the exterior. Therefore, to guarantee safety, we need to ensure that the agent stays in the non-negative region of $h(x)$, which can be represented as

$$
\mathcal{C} = \{x \in \mathbb{R}^n : h(x) \geq 0\} \tag{2.6}
$$

Figure 2.2: Graphic Model of CBF.

In [22], we have such a definition.

**Definition 1** $h(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ is continuous differentiable. The zero-superlevel set of $h$ is $\mathcal{C}$. For all $x \in \partial \mathcal{C}$, $\nabla h(x) \neq 0$. We have

$$
\sup_{u \in U} \frac{\partial h(x)^T}{\partial x} (f(x) + g(x)u) \geq -\alpha(h(x)) \tag{2.7}
$$

If there is an extended Class $K_{\infty}$ function $\alpha$ and there exists a set $\mathcal{D} \subset \mathbb{R}^n$ that equation (2.7) is satisfied when $\mathcal{C} \subset \mathcal{D}$, $h(x)$ is a CBF for all $x \in \mathcal{D}$. Any control rate satisfying Lipschitz continuity and suplevel’s constraint can be invariant to the safe set. $\dot{x} = f(x) + g(x)u$ is the control dynamics.

2.3.2 Create CBFs for STL tasks

According to Section 2.2.2, we know that the ”until” tasks can be divided into ”always” and ”eventually” tasks. In this section, we focus on creating CBFs for $G$ and $F$ tasks. Equation (2.2) shows that the STL formulas are based on time, therefore, we need to create a new CBF $b(x, t)$ that changes over time.
We choose to use a linear time function to construct CBFs [11]. The specific process is as follows:

- For the task \( \phi' \): \( G_{[a_k, b_k]} \phi \) or \( \phi' \): \( F_{[a_k, b_k]} \phi \), we can choose the time instant, which is described as

\[
    t^* = \begin{cases} 
        a_k & \text{if } G_{[a_k, b_k]} \phi \\
        b_k & \text{if } F_{[a_k, b_k]} \phi 
    \end{cases}
\]  

(2.8)

Therefore, we can ensure that for the "always" tasks, \( \phi' \) is satisfied for all \([a_k, b_k]\). For the "eventually" tasks, the satisfaction of \( \phi' \) will be achieved at least one time step between \([a_k, b_k]\).

- Choose the robustness value \( r_k \), i.e.,

\[
    r_k \in \begin{cases} 
        (0, \sup_{x \in \mathbb{R}^n} h_k(x)) & \text{if } t^*_k > 0 \\
        (0, h_k(x(0))] & \text{if } t^*_k = 0 
    \end{cases}
\]

(2.9)

where \( h_k(x) \) is the original CBF in equation (2.6).

- Define a piecewise linear function based on time, i.e.,

\[
    \gamma_k(t) := \begin{cases} 
        \frac{\gamma_{k,\infty} - \gamma_{k,0}}{t^*_k} t + \gamma_{k,0} & \text{if } t < t^*_k \\
        \gamma_{k,\infty} & \text{otherwise} 
    \end{cases}
\]

(2.10)

\[
    \gamma_{k,0} \in (-\infty, h_k(x(0))]
\]

(2.11)

\[
    \gamma_{k,\infty} \in \left( \max\left(r_k, \gamma_{k,0}\right), \sup_{x \in \mathbb{R}^n} h_k(x) \right)
\]

(2.12)

- The new CBF based on the STL formula can be defined as

\[
    b_k(x, t) := -\gamma_k(t) + h_k(x)
\]

(2.13)

First, the choice of \( \gamma_{k,0} \) will ensure that \( b_k(x(0), 0) > 0 \). Then, for \( t \geq t^*_k \), \( b_k(x, t) \leq -r_k + h_k(x) \) shows that \( b_k(x, t) \geq 0 \) when \( h_k(x) \geq r_k \). Therefore, we know that when \( t < t^*_k \), the new CBF will approach the target region at a linear rate. When \( t \geq t^*_k \), CBF will remain stable in the target region.

Thus, equation (2.7) can be rewritten as

\[
    \sup_{u \in U} \frac{\partial b(x, t)^T}{\partial x} (f(x) + g(x)u) + \frac{\partial b(x, t)^T}{\partial t} \geq -\alpha(b(x, t))
\]

(2.14)
Furthermore, if there are multiple tasks for each agent, we need to consider the conjunction of $n_b$ candidate CBFs $b_k(x, t), k = 1, 2, \ldots, n_b$. Authors in [19] show a Lemma with proof of CBF conjunction, which can be described as

$$-\ln \left( \sum_{i=1}^{n_b} \exp (-b_k(x, t)) \right) \leq \min_{i \in \{1, \ldots, n_b\}} b_k(x, t). \quad (2.15)$$

Therefore, the CBF for STL tasks can be rewritten as

$$b_{\text{all}}(x, t) = \min_{k \in \{1, \ldots, n_b\}} b_k(x, t) \approx -\ln \left( \sum_{k=1}^{n_b} s_k \exp (-b_k(x, t)) \right) \quad (2.16)$$

$$s_k = \begin{cases} 0 & t > b_k \quad \text{and} \quad k < n_b \\ 1 & \text{else} \end{cases} \quad (2.17)$$

where $s_k$ is a switch function based on time for different tasks. When $t > b_k$ and $k$-th task is not the last candidate CBF in the list, we can switch off the task when it finishes, which can prevent the agent from getting stuck in the old tasks. $b_{\text{all}}(x, t) \geq 0$ can ensure agent security and system stability.

### 2.4 Human-in-loop (HIL) Control

HIL control refers to a control system or process in which human operators actively participate and work collaboratively with an automated system to achieve a desired outcome. By incorporating human supervision and decision-making, the system can balance the results of human judgment and automated analysis. On the one hand, human feedback can be used to simplify the system and improve its performance and adaptability. On another hand, intelligent robots can quickly process repetitive tasks and get feasible solutions.

HIL control is applied in industrial automation, aviation, energy, and so on. Authors in [25] created a six-rotor UAV system where a human operator can supervise the UAV systems and manipulate them by sending command signals to the leader. In traffic scenarios, to ensure the safe operation of partially autonomous vehicles, human factors need to be taken into account. Paper [26] created experiments on human-vehicle interaction, considering different human factors, including reaction time, range of force, tactical behaviors, etc. Authors in [27] presented research on human-robot co-manipulation. In response to forces exerted by the human operator, the end effector of the robot manipulator will achieve a compliant behavior. Furthermore, HIL was also used in cyber-physical systems to cut energy waste [28].
There are many different ways for humans to participate in agent tasks, which can be described as follows.

1. In the low-level control process of the robots, humans can provide non-speech, joystick inputs to the system. Authors in [29] designed a mixed-initiative control system to mix human control input with robot input. An additive control function that measured the distance to unsafe regions was added as a coefficient of human input, which ensured safety despite possible human errors. Furthermore, robots can also adapt to human behavior and learn human preferences on parameters. Inverse reinforcement learning (IRL) can be used to learn human interest in optional tasks, by adjusting the priority of soft constraints, where paper [5] gave an example of implementation.

2. In some applications, humans can act as the full operator of an agent or group of agents, which are called human-driven robots. The goal of other intelligent robots is to cooperate with human-driven robots to complete tasks effectively. Author in [30] studied a cooperative system in which an agent is fully human-operated. By establishing different CBF constraints, other agents can maintain formation while the human-driven robot moves randomly, as well as ensure safety between them. Paper [31] considered a mixed-vehicle traffic scenario with autonomous and human-driven vehicles. By proposing an appropriate re-planning strategy in the low-level planner, the overlaps between the predicted trajectories of vehicles can be avoided.

3. In high-level planning, humans can give high-level instructions, such as "All follow me now", "One goes to area A", etc., to the intelligent agents at any time point. Agents need to integrate human tasks with existing tasks based on priority requirements, then obtain an efficient way to complete them. This form is the main focus of this paper.

2.5 Summary

This chapter mainly introduces the background knowledge of multi-agent coordination, task description, security control, and HIL design. In the following chapters, novel coordination approaches are designed based on background knowledge to deal with different kinds of tasks for multiple agents.

The main contributions include
• According to different communication types, including M2I and M2M communication, we design two coordination frameworks: distributed and centralized. The two framework designs have the same benchmark, which can be converted and integrated according to different scenarios. Thus, the frameworks have higher compatibility and applicability. At the same time, we can effectively avoid collisions between agents by designing CBF security constraints, which can achieve safety.

• Our method can establish STL formulas and CBF functions for different types of tasks. A task management system is proposed for multiple soft and hard tasks, which is effective for creating CBF constraints for the optimizations of different agents. In addition, considering HIL situations, our management system can convert high-level human orders into the corresponding STL formulas according to the command dictionary. We can also achieve task allocation to different agents according to the properties of human instructions (priority and agent number requirements). Task allocation is based on the combination of different indicators, including efficiency, trajectory smoothness, fairness, task redundancy, etc.

• For robot control, NMPC is used for complex nonlinear agent dynamics. Simulations and experiments are carried out to validate and evaluate the feasibility of our proposed methods.

The comparisons of our work with several existing investigations are shown in Table 2.2.

<table>
<thead>
<tr>
<th>Areas</th>
<th>Related Works</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAS Coordination</td>
<td>[9] [13]</td>
<td>Our work is compatible with different communication types.</td>
</tr>
<tr>
<td>Robot Control</td>
<td>[21]</td>
<td>Our work uses NMPC for nonlinear agent dynamics.</td>
</tr>
</tbody>
</table>
Chapter 3

Methods

The purpose of this chapter is to provide an overview of the research method used in this thesis. Section 3.1 describes the research process. Section 3.2 focuses on the data collection, analysis, and measurement techniques used for this research. Section 3.3 describes the experimental design including test environments and simulators. Section 3.4 explains the techniques used to evaluate the reliability and validity of the data collected.

3.1 Research Process

The research process in this thesis can be described into four steps, which can be shown as follows.

• **Step 1:** Literature study of the principles and applications of MAS, CBF, STL, HIL, and MPC.

• **Step 2:** Algorithm and framework designs successively on centralized and distributed coordination frameworks for STL tasks, as well as HIL model for applying high-level human commands to both frameworks.

• **Step 3:** Verification of proposed algorithms in a Matlab simulator to show their feasibility.

• **Step 4:** Experiments based on the CANOPIES ROS simulator to show the applicability of our proposed algorithms in agri-food scenarios.

Figure 3.1 shows a diagram of our research process, which includes theoretical research, simulations, and experiments.


3.2 Data Collection, Analysis and Measurements

This paper is mainly on developing coordination systems for STL tasks, where each agent completes control and achieves the tasks based on MPC. Therefore, we typically need the following data,

1. Model of the System Dynamics: The mathematical model that describes the dynamic behavior of each agent, which can be expressed as a set of differential equations. Both linear and nonlinear control models are considered in the simulations for mobile robots. We define parameters of the dynamics model, such as drift term, actuation effect, and time interval, according to the type of simulated robots.

2. Task Data: We set STL tasks according to common robot tasks and human commands, such as moving, staying in an area, following, etc. Considering the generality, the STL formulas we designed are mainly based on “eventually”, “always” and “until” tasks.

3. Control Objectives: Specify the desired control objectives to optimize based on different performance criteria, including control stability, trajectory smoothness, and so on.

4. Constraints: Define constraints on the system variables, such as control input constraints (speed and steering limits) and agent state constraints (initial conditions, CBF constraint, feasible space limits). Constraints ensure that the agent operates within safe and feasible areas.
5. Data on MPC Design: The prediction horizon is the duration of the prediction time of future system behavior. The control interval is the duration of applying the control actions to the agents.

We can analyze and measure the data and coordination results based on different performance indicators,

1. Feasibility of trajectories: It is based on whether there are collisions between trajectories, whether there are overlaps between trajectories and obstacles, whether trajectories meet dynamic requirements, and whether trajectories complete STL tasks.

2. CBF Performance: We can measure the CBF values of each task on each agent at different time steps to observe the execution of the task. When the values of CBFs meet the constraints, it means that the agents have been approaching the completion of the tasks in the correct direction.

3. Average computation time: Measure the time complexity of the coordination algorithms, which can provide requirements for the computing powers of the MEC or the onboard units.

4. Values of the Control Objectives: Measure the control solutions of the optimization problems using different criteria including control stability, trajectory smoothness, and so on.

3.3 Experimental Design

In this part, we will design different experiments for our coordination frameworks and HIL model. Furthermore, two different simulators for these experiments are introduced for experiments.

3.3.1 Test Environments

We consider multiple agent scenarios with $N$ agents. For the coordination frameworks, we need to design STL-based tasks for agents. Both centralized and distributed strategies are used to obtain MPC trajectories, then execute them based on designed dynamics. The details are shown in Table 3.1.

To test the HIL model, we need to design human commands with STL formulas and sent them to agents. By observing the updated coordination results, we can analyze the adaptability and feasibility of the model for HIL. The details are shown in Table 3.2.
### Table 3.1: Experiments for coordination frameworks

<table>
<thead>
<tr>
<th>Steps</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>Create agents and obstacles in the environment.</td>
</tr>
<tr>
<td>Step 2</td>
<td>Initialize hard tasks to agents based on environment for safety.</td>
</tr>
<tr>
<td>Step 3</td>
<td>Design soft tasks for agents for different goals.</td>
</tr>
<tr>
<td>Step 4</td>
<td>Create CBFs for both hard and soft tasks.</td>
</tr>
<tr>
<td>Step 5</td>
<td>Based on the communication type, use the proposed centralized or distributed strategy to obtain coordination results on each planning step.</td>
</tr>
<tr>
<td>Step 6</td>
<td>Control agents using agent dynamics, and evaluate the results based on criteria mentioned in Section 3.2.</td>
</tr>
</tbody>
</table>

### Table 3.2: Experiments for HIL system.

<table>
<thead>
<tr>
<th>Steps</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>Initialize the coordination system in Table 3.1.</td>
</tr>
<tr>
<td>Step 2</td>
<td>Design human commands and set them to agents at a specific time.</td>
</tr>
<tr>
<td>Step 3</td>
<td>Update STL tasks in the coordination strategy.</td>
</tr>
<tr>
<td>Step 4</td>
<td>Control agents using agent dynamics, and evaluate the results based on criteria mentioned in Section 3.2.</td>
</tr>
</tbody>
</table>

### 3.3.2 Simulators to be used

We use two different simulators for experiments, including the Matlab simulator and the CANOPIES ROS simulator.

#### 3.3.2.1 Matlab Simulator

MATLAB is a programming platform, it uses its own matrix-based language, which is suitable for algorithm simulation and verification. It can provide powerful mathematical modeling tools, which can easily construct and represent nonlinear dynamics and complete the planning, control, and simulation of MAS. It also has excellent numerical calculation and simulation capabilities, which can efficiently deal with complex STL tasks. With toolboxes of nonlinear optimization algorithms, such as “fmincon”, they can easily solve the NMPC problems in coordination. Furthermore, Matlab can also easily set and modify parameters, constraints, and objectives of coordination frameworks to evaluate and optimize system behavior.
In our experiments, we use Matlab simulation to test our proposed coordination frameworks and HIL model, as shown in Fig. 3.2.

![Figure 3.2: Example of Matlab simulator.](image)

### 3.3.2.2 CANOPIES ROS simulator

CANOPIES ROS simulator is based on Robot Operating System (ROS). First, it provides the model of tracked robots with robotic arms, which is suitable for harvesting and pruning tasks in agri-food areas. The dynamics of the robot is the nonlinear dynamics in equation (5.2). Actuators and sensors such as drive motors and inertial measurement units (IMU) are provided to control the robots precisely. The simulator also provides precise environment and location information, which is important for collision avoidance. Because ROS is a distributed system framework, the simulator can easily handle communication and collaboration in MAS. Each agent can function as an independent node and exchange information through the publish-subscriber communication provided by ROS.

We implemented the algorithms in the simulator based on Python. It provides rich solvers and libraries that can be used for modeling and solving NMPC, including "pyomo" and "ipopt" which we used in our implementation. Furthermore, the simulator provides a powerful visualization tool, which allows real-time debugging and monitoring of environmental changes and multi-agent movements.
3.4 Assessing reliability and validity of the data collected

In this part, we first need to analyze whether our coordination algorithms are reliable and effective, and then analyze whether the data collected from the algorithms can ensure the accuracy and reliability of the results.

3.4.1 Reliability and Validity of method

The proposed framework behaves consistently across multiple scenarios and input variations. First, in our frameworks, the agent dynamics are based on a general input-affine control system (4.1). Thus, in the experiment, we validate two commonly used robot types, (5.1) for 4WD Mecanum wheel robots, (5.2) for track-wheel mobile robots. For both dynamics, our frameworks can obtain feasible controls for all agents. Furthermore, we consider a variety of different tasks including "always", "eventually", and "until" tasks, as well as different human commands with different priorities. For all of our simulation tasks, the framework was successfully completed within the required time, which is consistent with the intended goals.

We also compare results from different algorithms and different platforms. By comparing the centralized and distributed solutions, we found that they both achieved efficient planning trajectories. We also used different simulation platforms in the experiment, including the Matlab platform and the Python-based ROS platform. In addition, sensitivity analysis can also be performed by changing the prediction length, objective function, etc.
3.4.2 Reliability and Validity of data

The data we collect is the data used by the coordinator and controllers of the CN and agents to execute. Thus, when the algorithm is reliable, we can also assume that the collected data is reliable, without being lost or destroyed. The first type of data is the trajectories combined with the past states of agents on different time steps. We can observe the completion status of the results by measuring whether the trajectories have collisions and whether they enter the task area within the required time range. Furthermore, CBF values of different tasks are important data, which should also be recorded to measure coordination safety and task completion. Different human instructions will also be analyzed to verify the reliability of the HIL model.
Methods
Chapter 4

MAS Coordination System based on STL tasks

This chapter will introduce our proposed coordination model based on STL, CBFs, and HIL. Our methods can be used in both centralized and distributed implementations, which will be presented in the following sections.

4.1 Problem Formulation

We consider $N$ agent in the scenarios. The nonlinear input-affine control system of agent $i$, $i \in \{1, 2, ..., N\}$ can be described as

$$\dot{x}_i(t) = f(x_i(t)) + g(x_i(t))u_i(t)$$  \hspace{1cm} (4.1)

where $x_i$ and $u_i$ is the current state and current control input. Function $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ represents the drift term in autonomous vector field and function $g : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$ represents actuation effect in control vector field. Both $f$ and $g$ functions are locally Lipschitz continuous.

For each agent $i$, we assign multiple ”always”, ”eventually,” or ”until” tasks to it. According to (2.5), we can split them into just ”always” and ”eventually” tasks. Depending on the properties of different tasks, we can reorganize these STL formulas into two lists, i.e.,

$$\Phi^i_S = \{\hat{\phi}^i_1, \hat{\phi}^i_2, ..., \hat{\phi}^i_{n_S}\}$$  \hspace{1cm} (4.2)

$$\Phi^i_H = \{\tilde{\phi}^i_1, \tilde{\phi}^i_2, ..., \tilde{\phi}^i_{n_H}\}$$  \hspace{1cm} (4.3)
with $\tilde{\phi}_k := G_{[a^i_k, b^i_k]} \phi | F_{[a^i_k, b^i_k]} \phi$, $\tilde{\phi}_k := G_{[a^i_k, b^i_k]} \phi$ and $\phi := \text{true} | \mu | \neg \mu$. List $\Phi_H^i$ refers to hard tasks, where their specifications should strictly be satisfied for safety, such as collision avoidance between agents and obstacles. Most of the hard tasks are $G$-tasks. List $\Phi_S^i$ refers to soft tasks, where their satisfactions are less stringent and can be violated or delayed, such as moving agents to different targets. For these tasks, the time range $[a^i_k, b^i_k]$ can be changed based on the task priority, for example, when we add some new tasks to the list.

Therefore, we have the STL formula in the form as

$$\phi''_i := \bigwedge_{k_1=1}^{n^H_2} \phi'_{k_1} \bigwedge_{k_2=1}^{n^H_2} \tilde{\phi}'_{k_2}$$

(4.4)

Our purpose is to create an online MPC solver to fulfill $\phi''_i, i = 1, 2, ..., N$ for each agent through coordination. We assume that the control interval is $\Delta t$ and the length of the predicted horizon is $N_p$. Therefore, to minimize our objective function $J$, the planner will use MPC to plan for the future control inputs $\tilde{U}_i$, which can obtain future states $\tilde{X}_i$, i.e.,

$$\tilde{U}_i = \{\tilde{u}_i(0|t_c), \tilde{u}_i(\Delta t|t_c), ..., \tilde{u}_i((N_p - 1)\Delta t|t_c)\}$$

(4.5)

$$\tilde{X}_i = \{\tilde{x}_i(0|t_c), \tilde{x}_i(\Delta t|t_c), ..., \tilde{x}_i(N_p\Delta t|t_c)\}$$

(4.6)

where $t_c$ is the current time and $\tilde{x}_i(0|t_c) = x_i(t_c)$.

### 4.2 CBF generator for Different Tasks

In this part, we will introduce the methods we proposed for generating CBFs for agents to achieve different tasks, including avoiding obstacles, preventing collisions with other possible agents within the communication range, reaching targets, following, and so on.

#### 4.2.1 Collision Avoidance for obstacles

These kinds of tasks are hard tasks for safety. When there are some static obstacles in the environment, we can decompose them into several moderately sized convex polygons and expand them into several circular areas. The goal then becomes to create a collision-free CBF function for each circular area, as shown in Fig. 4.1.
Therefore, we can use "always" STL formulas to describe this kind of task. We assume that \( p_{ob,k} = [x_{ob,k}, y_{ob,k}], k = 1, 2, \ldots, n_{ob} \) are the center points in Cartesian coordinates for \( n_{ob} \) circular obstacles. Then we can create an STL formula for each area in the form as

\[
\text{e}_h^k(x) = \|p - p_{ob,k}\| - (R_{ob,k} + R_{safe} + R_{red})
\]

\[
\text{e}_\phi^k = \bigwedge_{k=1}^{n_{ob}} G_{[a_k, b_k]}(\text{e}_h^k(x) \geq 0)
\]

where \( p \) is the Cartesian position of the agent in state \( x \), \( R_{ob,k} \) is the radius of the \( k \)-th circular obstacle, \( R_{safe} \) is the safe distance, \( R_{red} \) is the radius for redundancy. Therefore, based on equations (4.7) and (4.8), we can use the method mentioned in Section 2.3.2 to create \( n_{ob} \) CBF functions \( \text{e}_h^k(x, t) \) for static obstacle avoidance.

Figure 4.1: Safety region for static obstacle avoidance.

### 4.2.2 Collision Avoidance between Agents

In this part, we try to create CBFs to avoid collision with dynamic obstacles and other agents, which are also hard tasks. Since the two are using the same method, here we take the analysis of agent avoidance constraints as examples.

We set agent \( i \) as the ego agent and it needs to avoid agent \( j \). We assume that agent \( i \) has an inferior planning priority than \( j \), and they are within the communication limit to share information. Therefore, agent \( i \) can get the pre-planning trajectory \( \bar{X}_j \) of \( j \) through communication. As shown in Fig. 4.2, unlike the static obstacle equations (4.7) and (4.8), agent \( i \) needs to avoid a time corridor created by \( j \) between \( [t_c, t_c + N_p \Delta t] \), i.e.,

\[
\text{\bar{h}}^i(x_i, t) = \|p_i - p_j(t)\| - (R_{safe} + R_{red})
\]

\[
\text{\bar{\phi}}^i = G_{[t_c, t_c + N_p \Delta t]}(\text{\bar{h}}^i(x_i, t) \geq 0)
\]

where \( p_j(t) \) is the future predicted Cartesian positions of agent \( j \) obtained from \( \bar{X}_j \), and \( p_i \) is the position of agent \( i \).
However, because $\tilde{h}^i(x_i, t)$ is related to $t$, CBF $\tilde{b}^i(x_i, t) := -\gamma^i(t) + \tilde{h}^i(x_i, t)$ is no longer linear to time, which will cause enormous computation when calculating $\frac{\partial b}{\partial t}$ in equation (2.14). Therefore, We can discretize the time corridor of $j$ with the interval $\Delta t$ to set up the tasks, which can describe as

$$
\tilde{h}^i_k(x_i) = \| p_i - p_j(t_c + (k - 1)\Delta t) \| - (R_{safe} + R_{red}), \quad k = 1, \ldots, N_p \quad (4.11)
$$

Thus, the problem becomes solvable after discretizing. Similarly, we can create $N_p$ CBFs $\tilde{b}^i_k(x_i, t)$ for all predicted time steps to avoid agent $j$.

### 4.2.3 Reaching Targets

One of the most common soft tasks for mobile robots is staying or moving to a certain point or region at some time range. We can use the "always" or "eventually" STL formula to describe these kinds of tasks. We assume that $p_t = [x_t, y_t]$ is the position of the target point or the center position of the target region. Then, we can create an STL formula in the form as

$$
\hat{h}(x) = R_{reg} + R_{red} - \| p - p_t \| \quad (4.13)
$$

$$
\hat{\phi} = G_{[a, b]}(\hat{h}(x) \geq 0) \quad (4.14)
$$

where $R_{reg}$ is the radius for the target region ($R_{reg} = 0$ for reaching certain points). Similarly, we can create CBF function $\hat{b}(x, t)$ for reaching the target, which can be shown in Fig. 4.3.
4.2.4 CBFs for Other Tasks

There are many other different tasks, including following a human, ”until” tasks, and so on, which can also be solved by combining different CBFs mentioned in Section 4.2.1 and 4.2.3. The examples are shown as follows.

1. For the following task, considering the real-time human position is \( p_h(t_c) \), therefore the following task can be represented as a conjunction of \( F \) and \( G \) formulas, where \( F \) is for driving towards the human and \( G \) is for collision avoidance with the human, as shown in equation (4.15). \( R_{reg} > R_{safe} \) and the planning interval is \( \Delta t_p \). In addition, considering that the human is moving with time, we need to update the CBFs before each planning.

2. For an ”until” task, the agent needs to do task 1 until task 2 is finished. We use speed limitation \( u_{max} \) for task \( \phi_{task2} \) as an example, where it can be decomposed into a combination of \( G \) and \( F \) tasks, as shown in equation (4.16). Thus, we can create CBFs based on it.

\[
\text{Follow human : } \phi'(t_c) = \tilde{\phi}(t_c) \wedge \tilde{\phi}(t_c) = \bigwedge_{[a,b]} (p - p_h(t_c)) \leq R_{reg} + R_{red} \wedge \bigwedge_{[a,b]} (p - p_h(t_c)) \geq R_{safe} + R_{red}) \tag{4.15}
\]

\[
\text{Until task : } \tilde{\phi}' = \phi_{task1} U_{[a,b]} \phi_{task2} := \bigwedge_{[a,b]} \phi_{task1} \wedge \bigwedge_{[a,b]} \phi_{task2} \tag{4.16}
\]

Task 1: speed limit
4.3 Coordination Frameworks

In this part, we will propose both centralized and distributed frameworks for robot coordination. There are two main differences between the two methods. 1) Communication type: centralized framework is based on M2I communication and distributed framework is based on M2M communication. 2) Computing unit: for the first one, the optimization problem is solved on the high-level planner deployed on the CN, and for the second one, the solver is offloaded to the controller on each agent.

4.3.1 Centralized Coordination Framework

Figure 4.4: Structure of the centralized coordination framework.

The system structure is shown in Fig. 4.4. First, based on the normal tasks sent by the task allocator and some obstacle avoidance tasks obtained by the onboard cameras or lidars, each agent $i, i = 1, 2, ..., N$ will reform them into two task lists $\Phi^i_S$ and $\Phi^i_H$ for soft and hard tasks. Then, all the agents will transmit the current states $x_i(t_c)$, task lists, and previous trajectory $\hat{X}_i = \{x_i(0), x_i(\Delta t), ..., x_i(\lfloor t_c/\Delta t \rfloor \Delta t)\}$ to CN, where $\lfloor . \rfloor$ means rounding down.

At CN, after receiving the information sent by all the agents in communication ranges, the high-level planner will set a priority for each agent, which is decided by the ranking function $M$, i.e.,

$$M_i = M(x_i, \Phi^i_S, \Phi^i_H) = w_d || p_i(t_c) - p_h ||^2 + w_s \int_0^{t_c} \left( \frac{\partial^2 x_i(t)}{\partial t^2} \right)^2 dt$$  \hspace{1cm} (4.17)

The first term is the efficiency of the relative distance between the agent and the human, and the second term is the smoothness of the trajectory. The weighting parameters $w_d$ and $w_s$ are used to balance the importance of efficiency and smoothness.
where \( p_h \) is the target position of the first oncoming task in \( \Phi_i^S \). \( M \) consists of two parts. In the first part, we give superior priorities to agents closer to the target, which improves the efficiency of task completion. The second part is to give superior priorities to agents with smoother trajectories because it is easier to get directly to the target rather than making a lot of turns. We can use the finite difference method on \( \dot{X} \) to calculate the smoothness [32].

We can assign a place symbol \( 1 \leq \delta_i \leq N, \delta_i \in \mathbb{N}_+ \) to each agent by ordering the \( M \) values from smallest to largest. Importantly, lower \( \delta_i \) means superior priority. Then, we can save the order in a list \( \mathcal{L} \) at \( CN \), i.e.,

\[
\mathcal{L} = \{(i, M_i, \delta_i) \mid i = 1, 2, ..., N\}
\]

Therefore, based on the order, \( CN \) can add STL tasks for collision avoidance between agents. For agent \( i \), it needs to avoid multiple agents \( \{j \mid \delta_j < \delta_i\} \). Thus, we can add \( \text{len}(\{j \mid \delta_j < \delta_i\})N_p \) “always” tasks to \( \Phi_i^H \) based on equation (4.12), where \( \text{len}(.\) means the length of the list.

After that, according to Section 2.3.2, \( CN \) can create CBFs \( \tilde{b}_k^i(x_i, t), k = 1, 2, ..., n^i_S \) for \( \Phi_i^S \) and \( \tilde{b}_k^i(x_i, t), k = 1, 2, ..., n^i_H \) for \( \Phi_i^H \). It’s worth noting that we need to choose the values of \( \gamma_{k,0} \) and \( \gamma_{k,\infty} \) reasonably in equations (2.10) - (2.13). In order to avoid the influence of low-priority task \( k_2 \in \{1, ..., n^i_S\} \) on high-priority task \( k_1 \in \{1, ..., n^i_S\} \) where \( t_{k_1}^* \leq t_{k_2}^* \), \( t_{k_2}^* = \gamma_{k_2, \infty}, \gamma_{k_2,0}, \gamma_{k_2,\infty} \) need to satisfy \( -\gamma_{k_2}(t) \geq -\gamma_{k_1}(t), t \in [t_c, t_{k_1}^*] \). Then, based on equations (4.4) and (2.16), we can obtain the total CBFs \( \tilde{b}^{i}_{\text{all}}(x_i, t) \) and \( \tilde{b}^{i}_{\text{all}}(x_i, t) \).

The MPC optimization problem for agent \( i \) can be described as

\[
\mathcal{P}_1 : \min_{\dot{X}_i, e} \int_0^{N_p \Delta t} w_{u1} \| \tilde{u}(t|t_{c}) \|^2 + w_{u2} \left\| \frac{\partial \tilde{u}(t|t_{c})}{\partial t} \right\|^2 + w_e \| e \|^2 dt
\]

s.t. \( \dot{\bar{X}}_i(t|t_{c}) = f(\bar{X}_i(t|t_{c}))+g(\bar{X}_i(t|t_{c}))(\tilde{u}_i(t|t_{c}), a.e. \ 0, N_p \Delta t) \) \( (4.20) \)

\( \bar{X}_i(0|t_{c}) = x_i(t_c) \) \( (4.21) \)

\( \tilde{b}_{\text{all}}(\bar{X}_i, t + t_{c}) \geq -e, \ t \in [0, N_p \Delta t] \) \( (4.22) \)

\( \tilde{b}_{\text{all}}(\bar{X}_i, t + t_{c}) \geq 0, \ t \in [0, N_p \Delta t] \) \( (4.23) \)

\( \bar{X}_i(N_p \Delta t|t_{c}) \in C_P(t_c + N_p \Delta t) \) \( (4.24) \)

\( \bar{X}_i(t|t_{c}) \in X, \ t \in [0, N_p \Delta t] \) \( (4.25) \)

\( \tilde{u}_i(t|t_{c}) \in U, \ t \in [0, N_p \Delta t] \) \( (4.26) \)

\( e \in [0, \infty), \ t \in [0, N_p \Delta t] \) \( (4.27) \)
Equation (4.19) shows the objective function $J$ including control stability, acceleration smoothness, and CBF satisfaction. Equation (4.20) is the system dynamic based on (4.1). Equations (4.22) and (4.23) are the CBF constraints for soft and hard tasks, where soft CBF constraint has a relaxation factor $\epsilon$ and hard CBF constraint is strict to the non-negative region for safety. Equation (4.21) sets the initial state. Equation (4.24) is designed for terminal conditions to guarantee the recursive feasibility of MPC [21]. Equations (4.25) - (4.27) are for the value limitations of the state, control, and $\epsilon$ factor.

We can discretize $P_1$ to obtain the control inputs $U_i$ and future states $X_i$ within the predicted horizon, using the NMPC solver described in Appendix A. The details of the centralized coordination are given in Algorithm 1.

Algorithm 1 Centralized Coordination

1: Initialization. Set the number of agents $N$, predicted steps $N_p$, initial time $t_c = 0$, control interval $\Delta t$ and planning interval $\Delta t_p \leq N_p \Delta t$.
2: Initialization for each agent $i$, $i = 1, 2, ..., N$. Set initial state $x_i(t_c)$, previous trajectory $\hat{X}_i = []^T$ and initial task lists $\Phi_i^S$, $\Phi_i^H$.
3: repeat
4: Each agent updates $\Phi_i^S$, $\Phi_i^H$ based on the onboard sensors, task allocator, and human orders.
5: Each agent sends $x_i(t_c)$, $\hat{X}_i$, $\Phi_i^S$ and $\Phi_i^H$ to CN.
6: CN calculate $M_i$ based on (4.17) for all agents and set priority to them to obtain a ranking list $L$ (4.18).
7: for each $o \in [1, N]$ do
8: For agent $i \{i | \delta_i = o\}$, add collision avoidance tasks (4.12) for all agents $j \{j | \delta_j < o\}$ to $\Phi_H^i$.
9: For agent $i \{i | \delta_i = o\}$, create CBFs for $\Phi_S^i$ and $\Phi_H^i$.
10: Combine the CBFs to obtain $\hat{b}^i_{all}(x_i, t)$, $\hat{b}_{all}(x_i, t)$ based on (2.16).
11: Solve the optimization problem $P_1$.
12: Output: Predicted control and state matrices $\tilde{U}_i$ and $\tilde{X}_i$.
13: end for
14: Send back $\tilde{U}_i$, $\tilde{X}_i$ to each agent.
15: Control each agent during $n_p = \lfloor \frac{\Delta t_p}{\Delta t} \rfloor$ time steps with

$$u_i(t_c + (q - 1)\Delta t) \leftarrow \tilde{U}_i[q - 1], \quad q = 1, 2, ..., n_p$$ (4.28)

16: $t_c \leftarrow t_c + \Delta t_p$.
17: Update $x_i(t_c) \leftarrow \tilde{X}_i[n_p]$, $\hat{X}_i$, append($\tilde{X}_i[0 : (n_p - 1)]$).
18: until the coordination process is finished.
4.3.2 Distributed Coordination Framework

The system structure is shown in Fig. 4.5. As we can see, the computing task is offloaded into the controllers of the agents, and a communication graph is defined to describe the connections between agents. In this paper, we consider one of the most common cases, bidirectional and one-hop communication in M2M. Thus, the communication graph can be defined as an undirected graph shown in Fig. 4.6, and agent $i$ can only share information with its neighbors $j \in \mathcal{N}^i$, where out-neighbor set is the same with in-neighbor set in the undirected graph $\mathcal{N}_{\text{out}}^i = \mathcal{N}_{\text{in}}^i = \mathcal{N}^i$. 

![Figure 4.5: Structure of the distributed coordination framework.](image)

![Figure 4.6: Undirected graph for bidirectional communication.](image)
To avoid the double spending problem [33] caused by the establishment of multiple links on one node at the same time, there are two possible options.

1. **Method 1:** In initialization, we allocate the priorities $\delta_i$ from 1 to $N$, and never change them in the following plannings.

2. **Method 2:** Each agent $i$ store a ranking list $L^i$ in local, i.e.,

$$L^i = \{(z, M^i_z, \delta^i_z) | z = 1, 2, ..., N\}, \quad i = 1, 2, ..., N \quad (4.29)$$

Then, each agent $i$ needs to wait for list $L^i$ to become stable before local optimization at each time step.

According to the two methods, the interactions between agents can be divided into two progresses.

- **Ranking list update (for Method 2):** Agent $i$ can first update $M^i$ in its own list $L^i$ based on equation (4.17). Then the value $M^i$ will send to its neighbors in the communication graph. At the same time, agent $i$ will receive $\{M^j_j | j \in N^i\}$. Thus, we can update $M^i_j \leftarrow M^j_j$ in list $L^i$. After that, the order $\delta^i_z, z = 1, 2, ..., N$ can be reorganized by ranking the new $M^i_z, z = 1, 2, ..., N$ in agent $i$.

- **Planned trajectory transmission:** Agent $i$ will wait for the pre-planned trajectories from its superior-priority neighbors $\{\bar{X}_j | j \in N^i \text{ and } \delta^i_j < \delta^i_i\}$. Similarly, we can add STL tasks for agent avoidance to $\Phi^i_H$ and obtain total CBFs $\hat{b}^i_{\text{all}}(x, t)$ and $\tilde{b}^i_{\text{all}}(x, t)$. Through solving $\mathcal{P}_1$, we can get planned trajectory $\bar{X}_i$, which will send to its inferior-priority neighbors $\{j | j \in N^i \text{ and } \delta^i_j > \delta^i_i\}$.

The details of the distributed coordination are given in Algorithm 2. It’s worth emphasizing that lines 7 - 9 are specifically for Method 2, and we can remove the three lines for Method 1. Furthermore, we define two waiting times in the line 8 and 10, where $\Delta t_w$ is the estimated maximum waiting time for transmitting a signal to go from robot $i$ to all neighbors and back, and $\Delta t_c$ is a maximum calculation waiting for agents $j \{j \in N^i \text{ and } \delta^i_j < \delta^i_i\}$ to optimize their trajectories. However, when agent $i$ and agent $j$ cannot establish a handshake within the waiting period due to a communication error, we do not update $M^i_j$ and $M^j_i$ in line 8, or roll back them to the previous step and reorganize the priorities after line 10.
Algorithm 2 Distributed Coordination

1: Initialization. Set the number of agents $N$, predicted steps $N_p$, initial time $t_c = 0$, control interval $\Delta t$ and planning interval $\Delta t \leq \Delta t_p \leq N_p \Delta t$.

2: Initialization for each agent $i, i = 1, 2, \ldots, N$. Set initial state $x_i(t_c)$, previous trajectory $\hat{X}_i = [\cdot]^T$, initial task lists $\Phi_i^S, \Phi_i^H$, and ranking list $\mathcal{L}_i$ where we set $M_i^z = \text{Inf}$ and $\delta_i^z = z$.

3: Parallel For $i = 1$ to $N$ do in parallel

4: repeat

5: Agent $i$ updates $\Phi_i^S, \Phi_i^H$ based on the onboard sensors, task allocator, and human orders.

6: Agent $i$ builds communication with agents $j \in N^i$.

7: Agent $i$ waits $\Delta t_w$ for receiving $\{M_j^i|j \in N^i\}$, then map them in the local list $\{M_j^i \leftarrow M_j^i|j \in N^i\}$.

8: Agent $i$ reorganizes $\delta_i^z$ based on $M_j^z$, $z = 1, \ldots, N$.

9: Agent $i$ waits $\Delta t_w + \Delta t_c$ for receiving trajectories $\{\hat{X}_j|j \in N^i\}$ and $\delta_j^z < \delta_i^z$), and adds collision avoidance tasks to $\Phi_i^H$ based on (4.12).

10: Agent $i$ creates CBFs for lists $\Phi_i^S, \Phi_i^H$, then obtain $b_{\text{all}}^i(x_i, t), b_{\text{all}}^i(x_i, t)$ based on (2.16).

11: Solve the optimization problem $\mathcal{P}_1$.

12: Output: Predicted control and state matrices $\hat{U}_i$ and $\hat{X}_i$.

13: Agent $i$ sends its trajectory $\hat{X}_i$ to agents $j$, $\{j|j \in N^i\}$ and $\delta_j^z > \delta_i^z$.

14: Control agent $i$ during $n_p = \lceil \frac{\Delta t_p}{\Delta t} \rceil$ time steps based on (4.28).

15: $t_c \leftarrow t_c + \Delta t_p$.

16: Update $x_i(t_c) \leftarrow \hat{X}_i[n_p]$, $\hat{X}_i$. append($\hat{X}_i[0 : (n_p - 1)]$).

17: until the coordination process is finished.

18: End Parallel For

4.4 Human-in-loop (HIL) for Coordination

In HIL, we consider high-level human commands, which can be provided and sent to agents at any possible time. After receiving the commands, the agents will generate STL formulas, adjust their original tasks according to the requirements, and allocate human tasks at the coordination level. In this section, we will analyze the different types of human instructions and add them to our coordination framework.
4.4.1 HIL Model

A human command consists: Task Command (TC), Task Priority Command (PC), and Subject Command (SC). For example, in the command “All agents follow me now”, “follow me” is TC, “now” is PC, and “all agents” is SC.

For different TCs, we can create a command dictionary to map human tasks to different STL formulas, as shown in Table 4.1. The mapping of moving tasks can be shown in Section 4.2. Language models can be used to translate more informal English into specific STL tasks from our dictionary. We can also add other tasks depending on the application scenario, such as picking an item, bringing an item, and so on. In addition, combined with other systems, the human command dictionary can be extended. For example, perception-based tasks can be added based on hand gesture recognition systems [34] with speech-to-text modules. Similar to equations (4.2) and (4.3), we can also classify these human tasks into soft and hard task list \( \Psi_S \) and \( \Psi_H \), i.e.,

\[
\Psi_S = \{ \hat{\psi}_1, \hat{\psi}_2, \ldots, \hat{\psi}_{n_{\psi}} \} \tag{4.30}
\]

\[
\Psi_H = \{ \tilde{\psi}_1, \tilde{\psi}_2, \ldots, \tilde{\psi}_{n_{\phi}} \} \tag{4.31}
\]

with \( \hat{\psi}_k := G[a_k, b_k] \psi \mid F[a_k, b_k] \psi \), \( \tilde{\psi}_k := G[a_k, b_k] \psi \) and \( \psi := \text{true} | \mu | \neg \mu \).

For human soft tasks, PC includes ”now”, ”later” and ”a specific location”. Based on this requirement, we need to adjust the human soft task list \( \Psi_S \) and the original soft task list \( \Phi_S \) to achieve the adaptation between them, as shown in Table 4.2. The main idea is to delay execution time according to the insertion position of \( \Psi_S \). For hard tasks \( \Phi_H \) and \( \Psi_H \), they are usually unchangeable for safety. But for some static obstacles that remain constant in total time, their execution times need to be extended for extra activity according to the extension of the simulation time, i.e., \( b_k, b_k^h \leftarrow \Delta t_{\psi} + \Delta t_{\phi} \), where \( \Delta t_{\phi} \) and \( \Delta t_{\psi} \) are the total time for previous tasks and human tasks respectively.

There are three types of SCs, including ”All agents”, ”Agent \( i \in N_h \)”, and ”\( n_h \) agents”. ”All agents” refers to all the agents executing the same human tasks. ”Agent \( i \in N_h \)” means to assign specific agents for human tasks. ”\( n_h \) agents” only specifies the number of agents to perform tasks, but does not specify them. For these three different kinds of execution subjects, we need to develop different scheduling processes, which can be shown in Table 4.3. It is worth explaining that ”\( n_h \) agents” is a special situation in that we need to add human tasks to all agents first. Then, the task allocation between agents is completed according to the communication condition in coordination, which we will discuss in Section 4.4.2.
<table>
<thead>
<tr>
<th>Task Commands</th>
<th>STL Formulas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not do task</td>
<td>$\neg \psi_{\text{task}}$</td>
</tr>
<tr>
<td>Do task 1 and task 2</td>
<td>$\psi_{\text{task1}} \land \psi_{\text{task2}}$</td>
</tr>
<tr>
<td>Do task 1 until task 2 finished</td>
<td>$(4.16)$</td>
</tr>
<tr>
<td>Go to (GOAL)</td>
<td>$\psi_{\text{goto}[a,b]}$: $(4.14)$</td>
</tr>
<tr>
<td>Follow me</td>
<td>$\psi_{\text{follow}[a,b]}$: $(4.15)$</td>
</tr>
<tr>
<td>Pick (X)</td>
<td>$\psi_{\text{pick}}$</td>
</tr>
<tr>
<td>Bring (X)</td>
<td>$F_{[a_1,b_1]} \psi_{\text{pick}} \land \psi_{\text{follow}[a_2,b_2]}, b_1 \leq a_2$</td>
</tr>
</tbody>
</table>

Table 4.1: Dictionary of human task command (TC).

<table>
<thead>
<tr>
<th>Task priority Commands</th>
<th>Adjusting the task lists</th>
</tr>
</thead>
<tbody>
<tr>
<td>now</td>
<td>$\Phi'<em>S \leftarrow \Phi_S \left{ a_k = a_k + \Delta t</em>\psi, k \in {1, \ldots, n_S} \right}$</td>
</tr>
<tr>
<td></td>
<td>$b_k = b_k + \Delta t_\psi, k \in {1, \ldots, n_S}$</td>
</tr>
<tr>
<td></td>
<td>$\Delta t_\psi = \max \left{ b_k</td>
</tr>
<tr>
<td>later</td>
<td>$\Psi'<em>S \leftarrow \Psi_S \left{ a^h_k = a^h_k + \Delta t</em>\phi, k \in {1, \ldots, n^h_S} \right}$</td>
</tr>
<tr>
<td></td>
<td>$b^h_k = b^h_k + \Delta t_\phi, k \in {1, \ldots, n^h_S}$</td>
</tr>
<tr>
<td></td>
<td>$\Delta t_\phi = \max \left{ b^h_k</td>
</tr>
<tr>
<td>a specific location e.g. do human task</td>
<td>$\Psi'_S \leftarrow \Psi_S \left{ a^h_k = a^h_k + \tau_2, k \in {1, \ldots, n^h_S} \right}$</td>
</tr>
<tr>
<td>after task $\phi_{[\tau_1,\tau_2]}$</td>
<td>$b^h_k = b^h_k + \tau_2, k \in {1, \ldots, n^h_S}$</td>
</tr>
<tr>
<td></td>
<td>$\Phi'<em>S \leftarrow \Phi_S \left{ a_k = a_k + \tau_2 + \Delta t</em>\psi, k \in {k</td>
</tr>
<tr>
<td></td>
<td>$b_k = b_k + \tau_2 + \Delta t_\psi, k \in {k</td>
</tr>
<tr>
<td></td>
<td>$\Delta t_\psi = \max \left{ b_k</td>
</tr>
</tbody>
</table>

Table 4.2: Adjust the soft task lists based on task priority command (PC).

### 4.4.2 Task Allocation in Coordination

In this part, we will especially consider the ”$n_h$ agents” situations for human commands. For example, a human might ask for a number of robots to follow them without specifying a robot. The network of robots must decide which robots should fulfill this task while being as efficient as possible, as well as minimizing the impact on the original STL plan.

The solution we propose is based on a task allocation protocol. For the centralized coordination framework, it is running on the MEC server of CN, and for distributed framework, it is running on the background in a higher frequency thread of the robots’ computer. The allocation process is based on agent priority. Equation (4.17) can be rewritten to equation (4.32). The indicator consists of the following different components.
Subject Commands | Scheduling processes
--- | ---
All agents | (1) Update $\Phi^i_S, i \in \{1, ..., N\}, \Psi^i_S$ based on Table 4.2.
(2) $\Phi'^i_S = \Phi^i_S \cup \Psi^i_S, \Phi''_H = \Phi^i_H \cup \Psi^i_H, i \in \{1, ..., N\}.$
(3) Coordination based on $\Phi''^i_S$ and $\Phi''^i_H$ in Section 4.3.
Agent $i \in N_h$ | (1) Update $\Phi^i_S, i \in N_h, \Psi^i_S$ based on Table 4.2.
(2) $\Phi'^i_S = \Phi^i_S \cup \Psi^i_S, \Phi''_H = \Phi^i_H \cup \Psi^i_H, i \in N_h$
$\Phi'^i_S = \Phi^i_S, \Phi''_H = \Phi^i_H \cup \Psi^i_H, i \notin N_h.$
(3) Coordination based on $\Phi''^i_S$ and $\Phi''^i_H$ in Section 4.3.
n$_h$ agents | (1) Update $\Phi^i_S, i \in \{1, ..., N\}, \Psi^i_S$ based on Table 4.2.
(2) $\Phi'^i_S = \Phi^i_S \cup \Psi^i_S, \Phi''_H = \Phi^i_H \cup \Psi^i_H, i \in \{1, ..., N\}.$
(3) **Task Allocation** to $n_h$ agents in Section 4.3.

Table 4.3: Scheduling processes for different subject commands (SCs).

1. Human tasks are given superior priorities to agents that are closer to the target, which is to improve efficiency for human task completion. $p_t$ is the target point of the soft human task.

2. Human tasks are given superior priorities to agents that have smoother paths, which is to improve the trajectory comfort.

3. Considering the fairness of multiple agents, human commands will be assigned to agents with fewer tasks.

4. In addition, we want to assign human tasks to agents with longer allowable times on STL formulas which are called task redundancy. This gives the agents more room to perform human tasks without affecting other tasks. $t^*_k$ is defined in equation (2.8).

5. Finally, We want TCs to be assigned to agents that need less time to complete all tasks. $\Delta t^*_\phi$ and $\Delta t^*_\psi$ are the total time for completing previous tasks of agent $i$ and human tasks.

\[
M_i = w_d \| \mathbf{p}_i(t_c) - p_t \|^2 + w_s \int_0^{t_c} \left\| \frac{\partial^2 \mathbf{x}_i(t)}{\partial t^2} \right\|^2 \, dt + w_n [\text{len}(\Phi''^i_S) + \text{len}(\Phi''^i_H)] \\
- w_p \sum_{k, \hat{\phi}_k \in \Phi'^i_S} [t^*_k - a^*_k] + w_t (\Delta t^*_\phi + \Delta t^*_\psi)
\] (4.32)
In the centralized framework, by updating and sorting the new $M$ value from smallest to largest, we can obtain the ranking list $L$ in equation (4.18). For those agents of lower priority $\delta_i > n_h$, CN removes their soft human missions and remains the local tasks, i.e.,

\begin{align*}
\Phi''_i & \leftarrow \Phi'_i, & \delta_i > n_h \quad (4.33) \\
\Phi''_i & \leftarrow \Phi'_i \cup \Psi'_i, & \delta_i \leq n_h \quad (4.34)
\end{align*}

For distributed framework, the agent needs to exchange the $M$ value with its neighbors for updating its local ranking list $L_i$ in equation (4.29). Similarly, after that, each agent $i$ will decide whether to remove the human STL formulas based on its ranking on the local $L_i$, i.e.,

\begin{align*}
\Phi''_i & \leftarrow \Phi'_i, & \delta'_i > n_h \quad (4.35) \\
\Phi''_i & \leftarrow \Phi'_i \cup \Psi'_i, & \delta'_i \leq n_h \quad (4.36)
\end{align*}

Human hard tasks $\Psi_H$ should always be considered for all agents receiving the command. Therefore, we can successfully achieve high-level HIL in our coordination system.
MAS Coordination System based on STL tasks
Chapter 5

Results and Analysis

In this chapter, we present the numerical results of the proposed centralized and distributed coordination frameworks, as well as their applications on HIL commands. The simulations will be carried out on two platforms: (1) Matlab simulation to verify the feasibility and compare the performance of our proposed algorithms; (2) Simulation on the EU CANOPIES 3D Simulator by PaleBlue [1] to show the applications of our proposed frameworks on harvesting and pruning in agri-food areas.

5.1 Major results

In this section, the major results will be presented in three parts, including robot coordination, HIL tasks, and ROS-based experiments.

5.1.1 Robot Coordination

In this part, 3 agents are considered in the scenario with linear dynamics, i.e.,

\[
\begin{bmatrix}
  x_{i,new} \\
  y_{i,new}
\end{bmatrix} = \begin{bmatrix}
  1 & 0 \\
  0 & 1
\end{bmatrix} \begin{bmatrix}
  x_i \\
  y_i
\end{bmatrix} + \begin{bmatrix}
  dt & 0 \\
  0 & dt
\end{bmatrix} \begin{bmatrix}
  v_{x,i} \\
  v_{y,i}
\end{bmatrix}
\]

(5.1)

where \( x_i = p_i \). Three soft tasks and two hard tasks are assigned to each agent, as shown in Table 5.1. The first two soft tasks \( \hat{\phi}_1 \) and \( \hat{\phi}_2 \) are "Go to" tasks. The third soft task \( \hat{\phi}_3 \) is an "Until" task to set the maximum speed to half before completing \( \hat{\phi}_1 \). The first hard task \( e_{\phi_1} \) is to avoid a static obstacle that appeared from 0s to 10s and the second one \( e_{\phi_2} \) is to avoid collision with other agents.
Table 5.1: Simulation tasks for coordination frameworks.

Both centralized and distributed coordination frameworks are simulated. For the centralized one, we assume that 3 agents are connected at any time. For the distributed one, we assume that there is a communication range $R_c = 10$ m. When the distance between two agents is less than $R_c$, they can establish stable bidirectional communication. The main settings of coordination frameworks are shown in Table 5.2.

Table 5.2: Calibration of main parameters.

The coordination results for centralized and distributed frameworks are shown in Fig. 5.5 and 5.6. From the planned trajectories in the figures, we can see that both frameworks can obtain smooth feasible solutions to achieve all the STL tasks without any collisions with each other. From the CBF values in the figures, we can see that the CBFs of different STL formulas are all kept in the range of greater than or equal to 0 in the process of planning, which satisfies the constraint of equations (4.22) and (4.23).

Next, we will compare the two different frameworks. From Fig. 5.5f, CN will keep the CBF for agent avoidance (hard task 2) all the time in the centralized method, since the communication connections are always active. However, from distributed results in Fig. 5.6f, when the distance between agent $i$ and agent $j$ is larger than $R_c$, the agent $i$ will remove the avoidance CBF of agent $j$. 

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
Parameters & Description & Value \\
\hline
$N_p$ & Predicted horizon of MPC & 30 \\
\hline
$\Delta t$ & Control interval & 0.1s \\
\hline
$\Delta t_p$ & Planning interval (Centralized / Distributed) & 0.5s / 0.2s \\
\hline
$w_{u_1}, w_{u_2}, w_e$ & Weights of the objective function $J$ & 1, 10, 0.2s \\
\hline
$w_{d}, w_s, w_p, w_t$ & Weights of the ranking function $M$ & 1, 10, 1, 1, 1 \\
\hline
$R_{\text{safe}}$ & Safe distance for collision avoidance & 2m \\
\hline
$R_{\text{red}}$ & Radius for redundancy & 1m \\
\hline
$u_{\text{max}}$ & Maximum speed & 4m/s \\
\hline
\end{tabular}
\caption{Calibration of main parameters.}
\end{table}
Fig. 5.1a shows the cumulative distribution function (CDF) of average computation time per planning step. We can see that distributed strategy generally requires less computation time than the centralized one because the computing is offloaded to each agent. Fig. 5.1b compares the average J value (4.19) of the planning results on each time step between the two strategies. We can see that the results are similar, but in some time periods (3−8s), the centralized strategy had a lower J value. This is because the centralized strategy has a wider communication range and can synthesize information from other agents earlier to get a result closer to the global optimal.

![Graphs showing CDF and J value comparison](image)

(a) Average computing time  
(b) Average J value.

Figure 5.1: Performance Comparison.

### 5.1.2 HIL tasks

We also consider 3 agents in the scenario. Each agent has a pre-existing soft task \( \hat{\phi}_i^1 := F_{[5,10]} \left( \| p_i - p_{1i} \| \leq 1 \right) \) with \( p_{1i}^1 = [10, -10]^T \), \( p_{2i}^1 = [0, 0]^T \), \( p_{3i}^1 = [0, -11]^T \), and a pre-existing hard task \( \hat{\phi}_i^1 = G_{[0,T_{\text{max}}]} \left( \| p - [10, 0]^T \| \geq 4 \right) \). An extra human task is sent to the agents at \( t = 1 \). The distributed framework is used for coordination. We consider two different human commands,

1. **"All agents follow me later"**

   The STL formula we obtained from Table 4.1 is the same to equation (4.15), where human position \( p_h \) moving in x direction from 10−20s with speed 1m/s. The results are shown in Fig. 5.2. We can see that after receiving the HIL task at \( t = 1 \), agents first focused on the original task \( \hat{\phi}_i^1 \). Then, after finishing \( \hat{\phi}_i^1 \) in Fig. 5.3b, agents started to do the human task of "follow me". According to the simulation, our proposed framework can successfully complete the original tasks and human instruction.
2. "One agent go to \([10, 10]^T\) now": The STL formula we obtained from Table 4.1 is \(\hat{\psi} = F_{[5,10]} \left( \{\| p - [10, 10]^T \| \leq 1 \} \right)\). The results are shown in Fig. 5.3. We can see that after receiving the HIL task \(\hat{\psi}\) at \(t = 1\), all agents started to do it first. At \(t = 1.9\)s in Fig.5.3b, agent 2 and agent 3 built communication, then the task allocation protocol decided agent 2 to do the human task. Agent 3 removed \(\hat{\psi}\) from its soft task list and focus on \(\hat{\phi}_1^3\). At \(t = 4.8\)s in Fig.5.3b, agent 1 and agent 2 built communication, then the task allocation protocol decided agent 2 to do the human task. Similarly, agent 1 removed \(\hat{\psi}\) from its soft task list and focus on \(\hat{\phi}_1^1\). Finally, agent 2 completed the HIL task \(\hat{\psi}\) first and then completed its local task \(\hat{\phi}_2^2\). The simulation proves the feasibility of our task allocation protocol in Section 4.4.2.

Figure 5.2: Results on HIL command "All agents follow me later".

5.1.3 ROS-based Experiments

In this part, we will use the CANOPIES ROS Simulator to study a specific application for harvesting. Here, we need to consider a nonlinear dynamic model for mobile robots, i.e.,
where \( [x, y]^T \) is the robot position, and \( \theta \) is the yaw angle which describes the rotation around the z-axis. For control inputs, \( v \) is the forward velocity and \( \omega \) angular velocity around the z-axis.

In a vineyard scene, the most common task is to go to a specific spot \( p_{\text{goal}} \) to pick and place. This kind of soft task can be expressed as an "always" task \( F_{[a, b]} (\| p_i - p_{\text{goal}} \| \leq 1) \). In addition, since there are many poles in the grape arbor, hard tasks for obstacle avoidance need to be added to agents to ensure safety and avoid collisions. It can be represented as \( \tilde{\phi}_k : G_{[0, T_{\text{max}}]} (\| p_i - p_{\text{pole}, k} \| \geq 0.3), k = 1, 2, \ldots, n_{\text{pole}} \), where \( p_{\text{pole}, k} \) is the position of each pole and the value 0.3 is based on the radius of poles, as shown in Fig. 5.4. Furthermore, hard tasks for collision avoidance between agents also need to be considered.

Figure 5.4: Scenarios in the CANOPIES ROS Simulator.

5.2 Validity and Reliability Analysis

This part is mainly about the reliability of the data and analysis. First of all, our simulation is based on the classical dynamics models and NMPC controller. Then the tasks of the robots are defined in detail and reflected in the simulation. The CANOPIES ROS simulator based on real vineyard scenarios can restore the real scenes. Therefore, the data is considered reliable. From the above result analysis, both of the two different frameworks can complete the STL tasks and satisfy the CBF constraints. Similar results on two different simulators (Matlab, ROS system) can demonstrate the feasibility of the framework, which can also validate the effectiveness of our analysis.
Results and Analysis | 49

(a) Planned Trajectories 0 − 10s
(b) Planned Trajectories 10 − 20s

c) CBF for soft task 1
d) CBF for soft task 2

e) CBF for hard task 1
(f) CBF for hard task 2 and total CBF

Figure 5.5: Coordination results for centralized framework.
Figure 5.6: Coordination results for distributed framework.
Chapter 6

Conclusions and Future work

6.1 Conclusions

This work mainly deals with the coordination problem for MAS to solve multiple STL tasks for multiple robots. The main contributions can be shown as follows. (i) A CBF generator is created for different STL tasks including obstacle avoidance, collision avoidance between agents, reaching targets, following, and so on. (ii) A centralized coordination framework using CBF-based NMPC is formulated for M2I communication, which can generate efficient, safe, and smooth trajectories to solve STL tasks. (iii) A distributed coordination framework based on graph theory for M2M communication is generated to offload the computing tasks onto each agent. (iv) When high-level human commands are considered in the system, a HIL model is created to integrate human tasks into our coordination frameworks. A novel protocol is used for task allocation between agents.

Both Matlab and Python simulations are carried out. Numerical results show that (i) Both proposed centralized and distributed frameworks can obtain collision-free solutions for different ”eventually”, ”always” and ”until” tasks. (ii) Distributed framework can reduce the computation time of the system. (iii) Human tasks with different requirements on task priorities and task subjects can be solved using our proposed HIL model. (iv) Applications on harvesting and pruning scenarios are simulated using a CANOPIES ROS simulator, which further verifies the practicability and feasibility of our algorithms.

The presented frameworks along with algorithms are of value to the practical use of MAS. At the same time, the role of HIL in MAS is further explored in this work.


6.2 Limitations

The first limitation is the computation time. For the centralized coordination framework, the computational effort of CN increases with the number of participating agents. For the distributed framework, it achieves the purpose of reducing the computation time through distributed unloading. However, for a single agent, the computation amount increases with the number of tasks under consideration. Because as the number of STL formulas increases, we need to perform more CBF generating and stacking operations. Therefore, the number of agents and tasks carried by the frameworks is limited due to the limitation of the computing capacity of MECs and on-board units.

Another limitation is communication. Since our MPC-based planning is real-time, in order to ensure the two-way transmission quality of M2I and M2M communication, we need to have strict requirements on communication delay and jitter.

For STL robot and human tasks, this paper mainly considers moving-based tasks, such as going to point A, staying in a certain area, following, and so on. However, we do not give solutions to other types of tasks, such as grabbing, placing, etc.

This paper only simulates some common scenarios, robot tasks, and HIL tasks. In order to ensure the universality of the algorithms, we need to explore more special cases and tasks. Especially for HIL, more complex command forms, and couplings need to be considered.

6.3 Future work

This paper mainly includes three aspects: CBF generation for STL tasks, coordination frameworks, and high-level HIL. We provide algorithms and strategies for all three aspects. However, due to the breadth of the problem, in the second and third areas, further analysis and extension are needed to improve the scope of application of the frameworks. In this section, we will focus on some of the remaining issues that should be addressed in future work.

6.3.1 What has been left undone?

In our distributed system, we provided a coordination algorithm based on graph theory. But we also need to prove that the whole system can converge to a steady state, to increase the applicability of the algorithm.
6.3.1.1 Consensus in Distributed framework

In our proposed distributed framework, the collision avoidance between agents and the task allocation for HIL commands are based on agent priorities, which are defined by the ranking list $L^i, i = 1, 2, ..., N$ (4.29) saved in each agent. Consider a collaborative task, the closer agents are to fulfilling the task, the more connections will appear among agents. Thus, the exchange of $M$ values and the update of $L^i$ will occur more frequently. When it finally stabilizes, $L^i$ will reach a dynamic consensus [14]. It is important to develop a theoretical guarantee on the network decision in future work.

In addition, our distributed framework is based on ideal bidirectional communication. When we consider special communication jitter, the original communication undirected graph in Fig. 4.6 may become a directed graph. We need to prove that in this case, the coordination framework can still converge to a stable consensus.

6.3.1.2 Double spending problem in distributed framework

In Section 4.3.2, we introduce the double spending problem [33], when exchanging the priority order between agents. We offer two approaches: (i) pre-assigning an unchanging order label to each agent, (ii) first exchanging $M$ value, waiting for stability, and then local ranking to obtain $L^i$. The first method is inflexible, and the second one may result in a deadlock in the event of communication packet loss. Therefore, we need to continue an in-depth analysis of this problem to improve the robustness of the system, such as designing a local plan when there are communication errors and multiple-hop communication in the ad-hoc network.

6.3.1.3 Queue stability of tasks

For coordination systems with multiple tasks, maintaining the stability of task queue length can ensure the stability of task completion. In Section 4.4.2, we have taken fairness into account in the task allocation protocol. Considering that human instructions are continuously provided to agents in chronological order, we can further prove that this allocation strategy can keep the agent task queues $\Phi''_S^i$, $\Phi''_H^i$ stable, which allows the framework to remain balanced in a highly dynamic multitasking environment.
6.3.2 Next obvious things to be done

In particular, the author of this thesis wishes to point out that multiple types of human commands remain as a problem to be solved. Solving this problem is the next thing that should be done.

In Section 4.4.1, we modeled and summarized basic human commands. However, considering the diversity of human tasks and logic connections, more complex language logic should be considered when generating STL formulas. Therefore, we can combine it with the relevant research on language or gesture recognition systems to get a more reasonable model. Simulations based on human tasks need to be carried out on the CANOPIES ROS simulator to verify the feasibility of the proposed HIL model. Furthermore, we can also try to do experiments on real robots for real-world applications.

In addition, as for the use of CBF, this paper uses CBF with respect to time linearity. In future studies, we can consider other functions. For example, for low-priority tasks, we can adopt a function that declines slowly at the beginning and then accelerates. In this way, the influence of the low-priority CBFs on the gradient descent direction of the total CBF combination (2.16) can be decreased to a greater extent.

Another noteworthy research direction is to combine the centralized and distributed strategies in this paper into a double-level coordination framework, as shown in Fig. 6.1. Some high-level hard constraints such as collision avoidance between agents can be handed over to CN. CN can provide feasible tunnels that satisfy the hard tasks. Each agent completes soft tasks in its tunnel. In this way, the efficiency of task completion can be improved while the calculation time of coordination can be further reduced.

![Figure 6.1: Structure of double-level coordination framework.](image)
6.4 Reflections

One of the most important results of this thesis is the creation HIL model to apply high-level human commands to the coordination frameworks, as well as the experiments on the CANOPIES ROS simulator [1]. This work was supported by the ERC CoG LEAFHOUND, the EU CANOPIES project, the Knut and Alice Wallenberg Foundation (KAW), and the Digital Futures Smart Construction project, which are partly done by the SML lab at KTH.
Conclusions and Future work
References


Appendix A

Solver for NMPC

In this part, we modify the optimization problem $P_1$ and use NMPC solver to generate predicted control inputs and states for the future $N_p$ time steps.

A.1 Linear Dynamics

First, we consider a simplified linear dynamics model, i.e.,

$$x_{i,new} = Ax_i + Bu_i$$  \hspace{1cm} (A.1)

The predicted control inputs and states at $t_c$ can be represented as

$$
\begin{bmatrix}
\tilde{x}_i(\Delta t|t_c) \\
\tilde{x}_i(2\Delta t|t_c) \\
\vdots \\
\tilde{x}_i(N_p\Delta t|t_c)
\end{bmatrix} =
\begin{bmatrix}
A & 0 & \ldots & 0 \\
A^2 & AB & B & \ldots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
A^{N_p-1}B & A^{N_p-2}B & \ldots & B & \ldots
\end{bmatrix}
\begin{bmatrix}
x_i(t_c) \\
x_i(t_c) \\
\vdots \\
x_i(t_c)
\end{bmatrix}
+ 
\begin{bmatrix}
0 \\
0 \\
\vdots \\
0
\end{bmatrix}
\begin{bmatrix}
\tilde{u}_i(0|t_c) \\
\tilde{u}_i(\Delta t|t_c) \\
\vdots \\
\tilde{u}_i((N_p-1)\Delta t|t_c)
\end{bmatrix}

$$  \hspace{1cm} (A.2)

The objective function in equation (4.19) can be discretized and thus be rewritten as

$$J = \sum_{q=0}^{N_p-1} w_{u_1} \Delta t \|\tilde{u}_i(q\Delta t|t_c)\|^2 + \sum_{q=1}^{N_p-1} w_{u_2} \|\tilde{u}_i((q-1)\Delta t|t_c) - u_i(t_c - \Delta t)\|^2$$  \hspace{1cm} (A.3)
Then we can apply equation (A.2) to equation (A.3), where $J$ can expressed as quadratic form, i.e.,

$$
U_i^T \begin{bmatrix}
(w_{u_1} + 2w_{u_2})R & -w_{u_2}R & (w_{u_1} + 2w_{u_2})R & -w_{u_2}R & \cdots & (w_{u_1} + 2w_{u_2})R & -w_{u_2}R \\
-w_{u_2}R & (w_{u_1} + 2w_{u_2})R & -w_{u_2}R & \cdots & \cdots & (w_{u_1} + 2w_{u_2})R & -w_{u_2}R \\
-w_{u_2}R & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\end{bmatrix} U_i
$$

$$
-2w_{u_2} \begin{bmatrix} u_i(t_c - \Delta t) \\
0 \\
\vdots \\
0 \\
\end{bmatrix}^T U_i + w_i E^T E + \text{const}
$$

where $R$ is the cost coefficient matrix for control. We can set it to $I_{n_c \times n_c}$ where $n_c$ is the length of $u_i$. $E = [\epsilon_1, \epsilon_2, \ldots, \epsilon_{N_p}]$ is the matrix for CBF relaxation.

Based on equations (2.10) and (2.13), the nonlinear constraint (4.22) can be described as matrix form, i.e.,

$$
- \ln \left( \sum_{k=1}^{n_s} \hat{s}_k \exp \left( -\hat{\mathbf{B}}_k(\mathbf{X}_i, t) \right) \right) \geq \begin{bmatrix} \epsilon_1 \\
\epsilon_2 \\
\vdots \\
\epsilon_{N_p} \end{bmatrix} = \mathbf{E} \tag{A.5}
$$

$$
- \ln \left( \sum_{k=1}^{n_{s'}} \tilde{s}_k \exp \left( -\tilde{\mathbf{B}}_k(\mathbf{X}_i, t) \right) \right) \geq \begin{bmatrix} 0 \\
0 \\
\vdots \\
0 \end{bmatrix} \tag{A.6}
$$

$$
\hat{\mathbf{B}}_k(\mathbf{X}_i, t) := -\hat{\gamma}_k \begin{bmatrix} t_c + \Delta t \\
t_c + 2\Delta t \\
\vdots \\
t_c + N_p\Delta t \end{bmatrix} + \hat{h}_k(\mathbf{A}_t \mathbf{x}_i(t_c) + \mathbf{B}_t U_i) \tag{A.7}
$$

$$
\tilde{\mathbf{B}}_k(\mathbf{X}_i, t) := -\tilde{\gamma}_k \begin{bmatrix} t_c + \Delta t \\
t_c + 2\Delta t \\
\vdots \\
t_c + N_p\Delta t \end{bmatrix} + \tilde{h}_k(\mathbf{A}_t \mathbf{x}_i(t_c) + \mathbf{B}_t U_i) \tag{A.8}
$$
The constraints (4.24) - (4.26) can be described as linear constraints to $U_i$,

\[
\begin{bmatrix}
-x_{\text{max}} \\
\vdots \\
-x_{\text{max}} \\
\max \{-x_{\text{max}}, x_{C_{F1}}\}
\end{bmatrix} \leq A_i x_i(t_c) + B_i U_i \leq \begin{bmatrix}
x_{\text{max}} \\
\vdots \\
x_{\text{max}} \\
\min \{x_{\text{max}}, x_{C_{F2}}\}
\end{bmatrix}
\] (A.9)

\[
[-u_{\text{max}}^T, \ldots, -u_{\text{max}}^T]^T \leq U_i \leq [u_{\text{max}}^T, \ldots, u_{\text{max}}^T]^T
\] (A.10)

where $x_{\text{max}}$ and $u_{\text{max}}$ are value limitations for states and controls. $x_{C_{F1}}$ and $x_{C_{F2}}$ are the Lower and upper boundaries of $C_F$.

The optimization problem $P_1$ can be modified to

\[
\mathcal{P}_2 : \min_{U_i, E} \quad (A.4) \\
\text{s.t.} \quad (A.5)(A.6)(A.9)(A.10)
\]

$\mathcal{P}_2$ is a quadratic programming problem with both linear and nonlinear constraints, which can be solved by many off-the-shelf solvers such as ”fmincon” and ”ipopt”.

### A.2 Nonlinear Dynamics

In this part, we extend (A.1) to nonlinear dynamics (4.1). First, we need to make our state space equations linear, which can be described as

\[
x_{i,\text{new}} = x_i + f(x_i)dt + g(x_i)u_idt
\] (A.11)

When we consider $\Delta t$ as the control interval, it can be written as

\[
\bar{x}_i((q-1)\Delta t|t_c) = \bar{x}_i((q-1)\Delta t|t_c) + f(\bar{x}_i((q-1)\Delta t|t_c))dt + g(\bar{x}_i((q-1)\Delta t|t_c))\bar{u}_i((q-1)\Delta t|t_c)dt
\] (A.12)

with $q = 1, 2, \ldots, N_p$. Therefore, the constraints (A.5), (A.6), (A.9), and (A.10) can be divided step by step on $\bar{x}_i((q\Delta t|t_c)$ and $\bar{u}_i((q-1)\Delta t|t_c)$.

We can use ”pyomo” library to describe the constraints on each step. Similarly, the NMPC results can be obtained using ”ipopt” solver.