Autonomous mobile robots

Politecnico di Bari

Department of Electrical and Information Engineering
Industria 4.0 PhD Program

SSD: ING-INF/04 - Systems and Control Engineering
Final dissertation

AUTONOMOUS MOBILE ROBOTS

by Vito Andrea RACANELLI

Supervisors:
Prof. Saverio MASCOLO
Prof. Luca DE CICCO

Coordinator of PhD Program:
Prof. Caterina Ciminelli

Course n. XXXVII, 01/11/2021 - 31/10/2024





Department of Electrical and Information Engineering
Industria 4.0 PhD Program

SSD: ING-INF/04 - Systems and Control Engineering
Final dissertation

AUTONOMOUS MOBILE ROBOTS

---

by Vito Andrea RACANELLI

Referees:

Prof. Francesco DELLI PRISCOLI

Prof. Mario DI BERNARDO

Supervisors:

Prof. Saverio MASCOLO

Prof. Luca DE CICCO

Coordinator of PhD Program:
Prof. Caterina Ciminelli

Course n. XXXVII, 01/11/2021 - 31/10/2024
La ricerca è stata realizzata con il cofinanziamento dell'Unione europea - POC Puglia 2014-2020 – Asse X - Azione 10.4 "Interventi volti a promuovere la ricerca e per l'istruzione universitaria". Approvazione Avviso pubblico n. 1/POC/2021 "Dottorati di ricerca in Puglia XXXVII Ciclo".

Ai miei genitori, a mia nonna,

dalla testa dura e dal cuore tenero.

Abstract

This PhD thesis explores the challenges and solutions at the intersection of autonomous mobile robots (AMRs) and real-time video streaming, aiming to improve both robot control and communication systems. It begins by digging into the control strategies that drive AMRs, particularly focusing on Model Predictive Control (MPC) and Dynamic Programming (DP) to optimize decision-making and navigation. The research then shifts to industrial vehicles, proposing a fault-tolerant system designed to detect encoder sensor failures during motor speed control, which improves safety and reliability in real-world applications. The thesis also investigates the use of drones for tasks like surveillance and rescue, addressing the challenge of maintaining high-quality video streams despite fluctuating network conditions. A novel approach using Nonlinear MPC is introduced to optimize drone altitudes, ensuring clear video transmission and efficient altitude control. Finally, the thesis presents an adaptive bitrate algorithm, RT-MPC, to enhance video quality in real time by adjusting to network fluctuations. This work bridges the gap between autonomous robotics and video streaming, providing practical solutions for both fields.

Abstract (italiano)

Questa tesi di dottorato esplora le sfide e le soluzioni all'intersezione tra i robot mobili autonomi (AMR) e lo streaming video in tempo reale, con l'obiettivo di migliorare sia il controllo dei robot sia i sistemi di comunicazione. Partendo dalle strategie di controllo che guidano gli AMR, concentrandosi in particolare sul Model Predictive Control (MPC) e sulla programmazione dinamica (DP) per ottimizzare il processo decisionale e la navigazione, la ricerca si sposta poi sui veicoli industriali e propone un sistema fault-tolerant progettato per rilevare i guasti dei sensori di velocità durante il moto del veicolo. La tesi analizza anche l'uso dei droni per compiti come la sorveglianza e il soccorso, affrontando la sfida di mantenere flussi video di alta qualità nonostante le fluttuazioni delle condizioni di rete. Viene introdotto un nuovo approccio che utilizza un MPC Controller non lineare per ottimizzare le altitudini dei droni, garantendo una trasmissione video chiara e una gestione efficiente dei dati. Infine, la tesi presenta un algoritmo di bitrate adattivo denominato RT-MPC, per migliorare la qualità video in tempo reale adattandosi alle fluttuazioni della rete. Questo lavoro contribuisce a colmare il divario tra la robotica autonoma e lo streaming video, fornendo soluzioni pratiche per entrambi i campi.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Autonomous navigation of mobile robots assumes a critically important role in many areas such as autonomous cars, robots for logistic domains and exploratory robots. The progressive automation of manufacturing, industrial and military processes has led these systems to increasingly operate in irregular and unstructured environments, which requires a high degree of flexibility in perception, motion and control. In recent decades, significant milestones have been achieved in mobile robot technologies. However, autonomous navigation in unstructured environments is still an open issue.

In the context of logistics systems, one of the most important issues to consider is the movement of materials within industrial environments. Despite the high performance of static material handling technologies (e.g., rollers or chain conveyors), the vast majority of industrial applications rely on common pallet trucks or forklifts as transport systems. The impact of mobile robotics within warehouses and industries is set to accelerate over the next five years. According to [4] by research and consulting firm LogisticsIQ, the automation market in logistics is set to double from $13 billion in 2018 to $27 billion in 2025. There are

many reasons for this: in addition to cost aspects, one of the main advantages is the unparalleled flexibility with which these simple systems can be integrated into existing or highly dynamic environments. The extension of the advantages of these simple forklifts, through the use of available technologies for industrial automation, has given rise to Autonomous Guided Vehicles (AGVs), which are highly reliable and useful means of reducing logistics costs and represent the most advanced automated type of vehicle for logistics environments. With more robots on the horizon, issues related to the safety of autonomous navigation will become increasingly important.

In the field of exploration, we can make a distinction between purely exploratory robots, whose main task is the discovery of a new environment (e.g., Mars Rover) and reconnaissance robots used for special tasks (e.g., Search and Rescue robots used in disasters). For the latter type, Robin Murphy, professor of computer science and engineering at Texas A&M University, says that "Real disasters are rare and each one is different. Robots will never be used exactly as thought out in the design phase and will continue to bring up new bottlenecks and problems that need to be solved." [5] For these systems, the open problems are divided into: sensory information processing, full mobility control, and robot manipulation skills. Solving them would lead to these robots being able to be used in all (or almost all) real-world situations.

The control of autonomous robots can be divided into the control of the platform (which falls under the domain of automatic controls) and the control of its activities (which falls under the domain of artificial intelligence). Thus, the development of autonomous mobile robots requires the use of both automatic control techniques and artificial intelligence techniques. The high mechanical complexity, of all modes of operation and the various requirements due to the environment

pose major challenges for automation.

In recent years, video systems have become an integral part of modern robotic platforms, playing a critical role in expanding their capabilities across a range of applications. From autonomous mobile robots (AMRs) to unmanned aerial vehicles (UAVs), video technology enables robots to perceive, interpret, and respond to their surroundings with a level of detail and accuracy that was previously unattainable. By processing visual information from RGB cameras, infrared sensors, or depth-sensing (RGB-D) cameras, robots can identify features in their environment, track moving objects, and make decisions based on their surroundings. For instance, in autonomous vehicles, video feeds combined with sensor fusion techniques help create comprehensive situational awareness, allowing the robot to navigate safely and effectively.

The integration of advanced video processing techniques, such as computer vision and machine learning, has further enhanced the autonomy of robotic systems. Algorithms like convolutional neural networks (CNNs) can analyze video streams in real time, allowing robots to recognize complex patterns and respond to dynamic environments. This capability is essential in tasks like autonomous inspection, where robots use video analysis to detect anomalies in industrial settings or in search-and-rescue operations, where UAVs equipped with video cameras scan large areas to locate survivors.

Video systems also facilitate remote monitoring and control [6], which is particularly valuable in hazardous or inaccessible environments. By transmitting video feeds to a ground control station (GCS) or a remote operator, robots can perform tasks in environments that are dangerous or difficult for humans to access, such as deep-sea exploration or disaster zones.

While video systems significantly extend the functionality of robots, they also

present unique challenges. High-resolution video data requires substantial bandwidth for transmission [c5], and maintaining video quality becomes difficult in environments with limited or fluctuating network connectivity. Adaptive bitrate streaming and video compression techniques have been developed to address these issues, allowing robots to adjust video quality in real-time based on available bandwidth. This adaptability is essential for maintaining operational reliability, particularly in mission-critical applications where consistent video feedback is necessary.

This thesis explores new ways to improve control, safety, and performance in autonomous mobile robots and drones, with additional applications in video streaming. At its core, the research focuses on using Model Predictive Control (MPC) and Dynamic Programming (DP) to make these systems more effective and reliable in real-world settings.

One part of the work introduces a fault-tolerant control system designed for industrial vehicles. Unlike traditional systems that mostly focus on motor or torque control, this solution emphasizes detecting and handling sensor faults—a critical safety issue that's often overlooked. By meeting European safety standards, this approach enhances the safety and dependability of industrial vehicles.

In another section, the thesis presents a collaborative control strategy for swarms of drones, particularly useful in complex tasks like search and rescue. To support real-time video streaming from multiple drones to a Ground Control Station, an adaptive strategy is proposed that adjusts drone altitude to maintain video quality even when network conditions vary. This provides consistent, high-quality footage that can be crucial for real-time decision-making in emergency scenarios.

The final contribution is *Real-Time MPC (RT-MPC)*, a new adaptive bitrate

algorithm for video streaming applications based on Model Predictive Control and Dynamic Programming. RT-MPC is designed to adjust video quality on the fly, ensuring smooth playback and a better user experience even when network speeds fluctuate.

All these advancements bring new levels of safety, efficiency, and user experience to autonomous systems and streaming technology, making them more capable and reliable across a wide range of practical applications.

1.1 Model Predictive Control (MPC)

Model Predictive Control (MPC) is a control strategy used to optimize decision-making processes by predicting and optimizing the behavior of a system over a future time horizon. MPC leverages a mathematical model of the system to predict future states, then computes an optimal control sequence by minimizing a specified cost function subject to constraints.

At each time step t, the controller uses the current state of the system to predict its future behavior over a finite prediction horizon H_P. It calculates an optimal sequence of control actions $\mathbf{u} = \{u_t, u_{t+1}, \ldots, u_{t+H_P-1}\}$ that minimizes a given objective function while satisfying system constraints. However, only the first control action u_t is implemented. At the next time step, the process repeats, using updated system states to re-compute the control sequence. This iterative process allows MPC to adapt to changing system dynamics and external disturbances.

1.1.1 Mathematical Formulation of MPC

Let x_t denote the system state at time t, and let u_t represent the control action applied at time t. MPC optimizes a sequence of control actions by minimizing a cost function J, which is generally formulated as follows:

$$J(\mathbf{u}, x_t) = \sum_{k=0}^{H_P-1} [\ell(x_{t+k}, u_{t+k}) + \ell_f(x_{t+H_P})] \tag{1.1}$$

where:

- $\ell(x_{t+k}, u_{t+k})$ is the stage cost function, representing the cost associated with the state x_{t+k} and control action u_{t+k} at each time step k within the prediction horizon.

- $\ell_f(x_{t+H_P})$ is the terminal cost function, representing the cost at the end of the prediction horizon to encourage certain end-state conditions.

- H_P is the length of the prediction horizon, which defines how far into the future MPC makes predictions.

The system dynamics are often represented by a discrete-time state-space model:

$$x_{t+1} = f(x_t, u_t) \tag{1.2}$$

where f is the system function that describes how the state x_t transitions to x_{t+1} under control action u_t. The control objective is to choose the control actions $\mathbf{u}$ such that the cost J is minimized, subject to system constraints.

1.1.2 Constraints in MPC

MPC typically operates under a set of constraints that represent physical and operational limitations of the system. These constraints include:

- **State Constraints:** The system state x_t must satisfy certain limits, such as safe operating conditions. State constraints can be represented as:

$$x_{\min} \leq x_t \leq x_{\max} \tag{1.3}$$

  where $x_{\min}$ and $x_{\max}$ define the minimum and maximum allowable states.

- **Control Constraints:** Control actions u_t must remain within feasible bounds to avoid overstressing the system. Control constraints are given by:

$$u_{\min} \leq u_t \leq u_{\max} \tag{1.4}$$

  where $u_{\min}$ and $u_{\max}$ are the lower and upper bounds on control actions.

- **Terminal Constraints:** At the end of the prediction horizon, additional constraints may apply to ensure desired end conditions. These terminal constraints can help stabilize the system and guide it toward a target state [7].

1.1.3 Optimization Problem in MPC

The MPC optimization problem can be formulated as a constrained optimization problem, which seeks to find the optimal control sequence $\mathbf{u}^* = \{u_t, u_{t+1}, \ldots, u_{t+H_P-1}\}$ that minimizes the cost function J while satisfying all constraints. This problem is represented as:

$$\begin{aligned} \min_{\mathbf{u}} \quad & J(\mathbf{u}, x_t) = \sum_{k=0}^{H_P-1} \left[\ell(x_{t+k}, u_{t+k}) + \ell_f(x_{t+H_P})\right] \\ \text{subject to} \quad & x_{t+k+1} = f(x_{t+k}, u_{t+k}), \quad k = 0, \ldots, H_P - 1 \\ & x_{\min} \leq x_{t+k} \leq x_{\max}, \quad k = 0, \ldots, H_P \\ & u_{\min} \leq u_{t+k} \leq u_{\max}, \quad k = 0, \ldots, H_P - 1 \end{aligned} \tag{1.5}$$

This optimization problem is typically solved at each time step t using numerical methods. Once the optimal sequence $\mathbf{u}^*$ is obtained, only the first control action u_t^* is applied to the system. At the next time step, the optimization is repeated with updated system states, making MPC a receding horizon control strategy.

1.2 Dynamic Programming

Dynamic Programming (DP) and MPC are similar in their capability to address challenging, multi-stage decision problems as optimization methods. On the other hand, DP is frequently employed in situations where decisions can span a long or indefinite horizon, with each overlapping subproblem only needing to be solved once, instead of recalculating decisions at every time step as in MPC. The DP approach involves decomposing complex problems into smaller, manageable subproblems and storing intermediate results, making it very effective for specific optimization problems.

DP is based on two fundamental principles: optimal substructure and overlapping subproblems. The concept of optimal substructure means that the best solution to a problem can be found by using the best solutions to its smaller parts. For example, if a problem involves identifying the most effective mix of individual

solutions, each of those individual solutions must be of top quality to attain the best overall outcome.

The second principle of DP sets it apart from methods like MPC that recompute the best path each time. DP is especially advantageous when smaller problems reappear frequently in the overarching problem. DP avoids unnecessary computations and boosts efficiency by saving subproblem results in a cache or table to prevent redundant calculations and decrease computational overhead.

DP typically operates in a *bottom-up* manner, where solutions to smaller subproblems are computed first and combined to form the solution to the full problem. Alternatively, it can be implemented in a *top-down* recursive approach with *memorization*, where intermediate results are stored as they are computed. This technique significantly reduces the computational complexity compared to brute-force methods, particularly for problems with exponential growth in solution space.

1.2.1 Mathematical Formulation of Dynamic Programming

Let $V(x)$ represent the value function for state x, denoting the best achievable outcome from that state onward. The recursive nature of DP is captured in the **Bellman equation**, which expresses $V(x)$ in terms of the values of subsequent states. For a minimization problem, the Bellman equation is defined as:

$$V(x) = \min_{u \in U} \left[g(x, u) + V(f(x, u)) \right] \tag{1.6}$$

where:

- $V(x)$ is the value function at state x,
- U is the set of available actions u at state x,

- $g(x, u)$ represents the immediate cost or reward of taking action u at state x,
- $f(x, u)$ denotes the state transition function, representing the next state from taking action u in state x.

The objective is to select a sequence of actions that minimizes (or maximizes) $V(x)$, forming an optimal policy. DP often employs a backward induction method, where the problem is solved recursively from the final state backward to the initial state, which contrasts with MPC's forward-predictive approach.

By eliminating redundant calculations through caching or memorization, DP reduces the computational complexity of solving problems, especially those that involve recursive subproblems. This approach enables DP to solve complex problems more efficiently than traditional methods. Additionally, DP systematically evaluates all potential solutions in cases where optimal substructure is present, thereby ensuring that it can find globally optimal solutions.

1.3 Scientific contributions

The research activity carried out during the PhD has led to the following publications:

International Journals

- Gioacchino Manfredi, Vito Andrea Racanelli, Luca De Cicco, Saverio Mascolo, *Live Streaming Synchronisation Using Event-triggered Consensus Control*, IEEE Transaction on Control of Network Systems (accepted).

International Conferences

- Walter Brescia, Giuseppe Roberto, Vito Andrea Racanelli, Saverio Mascolo, Luca De Cicco, *Point2Depth: a GAN-based Contrastive Learning Approach for mmWave Point Clouds to Depth Images Transformation*, Proc. 31st Mediterranean Conference on Control and Automation (MED2023), Limassol, Cyprus, June 26 – 29, 2023

- Gioacchino Manfredi, Vito Andrea Racanelli, Luca De Cicco and Saverio Mascolo, *LSTM-Based Viewport Prediction for Immersive Video Systems*, Proc. of MedComNet 2023, Ponza, Italy, 13-15 June 2023

- Vito Andrea Racanelli,Saverio Mascolo, *Safe and Fault Tolerant Control of Industrial Differential Drive Vehicles*, Proc. of 12th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes, Ferrara, Italy, May 2024

- Mohammad Amin Rezaei, Gioacchino Manfredi, Vito Andrea Racanelli, Saverio Mascolo, Luca De Cicco, *Decentralized Control of UAV Swarms for*

Bandwidth-Aware Video Surveillance Using NMPC, Proc. of International Conference on Unmanned Aircraft Systems (ICUAS 2024), Chania, Greece, June 2024

- Vito Andrea Racanelli, Gioacchino Manfredi, Luca De Cicco, Saverio Mascolo, *Real-Time MPC for Adaptive Video Streaming*, IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, USA, January 2025, (accepted).

- Mohammad Amin Rezaei, Gioacchino Manfredi, Vito Andrea Racanelli, Luca De Cicco, Saverio Mascolo *An Online Path Planner for Bandwidth-aware Aerial Camera Networks*, American Control Conference (ACC), Denver, USA, July 8-10, 2025, (submitted).

1.4 Thesis Outline

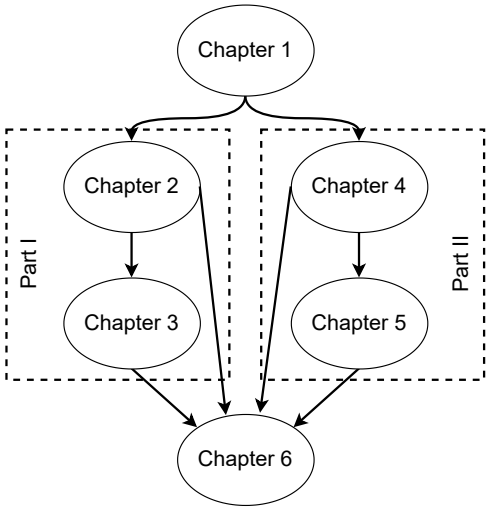

Figure 1.1: Chapter organization of the thesis

The thesis is structured as follows:

- Chapter 1 introduces Autonomous Mobile Robots (AMRs) with a brief historical overview. It focuses particularly on the key techniques of Model Predictive Control (MPC) and Dynamic Programming (DP), which are central to the control and optimization of these systems.

- Chapter 2 shows a fault-tolerant control system for industrial vehicles, focusing on detecting failures during speed control. While most research targets motor and torque control, this work addresses a critical gap in sensor failure detection. Experimental results show the system effectively handles sensor faults, keeping the vehicle safe and operational. The approach meets European safety standards, contributing to improved safety in industrial vehicles.

- Chapter 3 focuses on the use of UAVs, particularly multi-rotor drones, for

tasks like surveillance, search and rescue, and firefighting. It highlights the benefits of UAV swarms, including increased coverage and efficiency, especially in complex tasks like collaborative mapping and sensing. The chapter emphasizes the need for autonomous capabilities in each drone, such as path planning and obstacle avoidance, along with coordinated control strategies to ensure smooth operation within the swarm. The main challenge discussed is the real-time video streaming from drones to a Ground Control Station (GCS), where maintaining video quality in the presence of fluctuating network bandwidth is critical. To address this, the authors propose adjusting the drones' altitude to improve video quality while managing the overlap needed for video stitching. To solve these issues, the chapter introduces a Nonlinear Model Predictive Control (NMPC) framework.

- Chapter 4 provides an overview of video streaming technology, with a special focus on Quality of Service (QoS) and Quality of Experience (QoE). It also includes an analysis of different video quality metrics, highlighting their importance in evaluating streaming performance.

- Chapter 5 presents RT-MPC, an innovative Adaptive Bitrate (ABR) algorithm designed to improve video quality when streaming over time varying bandwidth channels. RT-MPC efficiently adjusts video quality in real time to increase the user QoE. It reduces the complexity of traditional Model Predictive Control (MPC) through dynamic programming and encoding constraints, making it suitable for real-time use without compromising performance.

- Chapter 6 provides discussions and conclusive remarks along with possible future research directions.

Part I

Mobile Robotics

Chapter 2

On Safety Speed Controller for Differential Drive Vehicles

In this chapter, the practical importance of developing robust and fault-tolerant controllers is analyzed. Specifically, the problem is set for the case of industrial vehicles with differential traction, which are inherently more hazardous in the event of faults [8].

2.1 Background

In manufacturing and logistics, vehicles are essential for carrying out tasks and transporting loads. These vehicles can vary significantly in weight, ranging from a few hundred kilograms to several tens of tonnes, depending on their intended use. Such variations in size and mass mean that these vehicles can pose substantial safety risks during operation, especially when maneuvering around human operators.

Most incidents involving industrial vehicles are due to operator errors, such as

distraction or carelessness. However, a smaller proportion of accidents are caused by failures in sensors, actuators, or control systems installed on these vehicles [9]. To mitigate these risks, the European Union Directive 2006/42/EC [10] and various international standards (e.g., UNI 280) set mandatory safety requirements for manufacturers of such machinery.

The required safety levels are directly correlated with the potential hazards posed by these vehicles. The higher the potential for damage in case of a vehicle failure, the more stringent the required safety measures. This is assessed using Safety Integrity Levels (SIL), [11], which quantify a safety system's reliability based on the probability of failure on demand (PFD). There are four levels—SIL 1 through SIL 4—with higher SIL levels indicating a lower probability of failure and a higher assurance of safety. However, increasing the SIL level generally results in greater costs and system complexity. It is essential for designers to integrate these safety requirements into the vehicle's design process to ensure safety without significantly compromising functionality.

Vehicles in industrial environments are primarily electrically powered and typically operate with an open-loop control system, where an operator's input (via a joystick, lever, or pedal) directly influences the motor's drive signal. While straightforward, this method can be less effective for controlling vehicle maneuvers. For example, the same control input may produce different vehicle behaviors when moving uphill versus downhill. This limitation can be addressed by implementing a feedback-controlled speed system using speed sensors (e.g., encoders) and well-established PID controllers [12]. While designing such systems is relatively straightforward, ensuring their safety and reliability is challenging. As the complexity of a control system increases, so does the likelihood of failures. Thus, it is critical to design these systems with a thorough understanding of potential

failure risks, allowing for the development of effective countermeasures [13, 14].

The literature on fault-tolerant speed control systems is extensive. For example, [15] provides a detailed survey of various sectors utilizing these systems. Despite significant advances in fault-tolerant control techniques, there has been less focus on managing sensor failures, especially those affecting rotary speed measurements.

Research in this area includes [16], which explores sensor fault detection within a reconfigurable direct torque control system applied to an electric vehicle powered by an induction motor. This study focuses on detecting faults in current, voltage, and speed sensors, followed by the implementation of fault-tolerant control measures to maintain vehicle operation. In [17] a posterior reliability voting algorithm is introduced to evaluate the integrity of an encoder, recalibrating the weights between encoder-measured velocity and model-based estimated velocity. In [18] a fuzzy logic for motor state diagnosis and fault detection is proposed. Moreover, [19] presents an active fault-tolerant control method for 4WD electric vehicles, based on an unmatched disturbance observer and an adaptive sliding mode control. This approach ensures stability and tracking performance even with actuator faults, as demonstrated through hardware-in-the-loop simulations. The proposed model emphasizes robustness and safety, enabling continued vehicle operation under various failure scenarios.

This body of work highlights the importance of addressing sensor and actuator reliability in the design of control systems for industrial vehicles, particularly given the stringent safety standards required by both regulatory frameworks and industry practice.

2.2 Safety regulations and risk metrics

2.2.1 Risk reduction metrics

SIL and PL are both risk reduction metrics used in functional safety standards, especially in machinery safety, to evaluate and quantify the reliability of safety functions. They help in setting and achieving appropriate safety performance levels, ensuring machinery operates safely and reduces risks to an acceptable level.

2.2.1.1 SIL (Safety Integrity Levels)

SIL is defined in IEC 61508 [11] (and referenced in EN 62061 [20] for machinery) and stands for Safety Integrity Level. It quantifies the probability of failure of a safety function in a safety-related system. SIL levels range from SIL 1 to SIL 4, with SIL 4 providing the highest level of risk reduction and reliability. Higher SIL levels indicate greater safety reliability but also require more robust engineering, testing, and redundancy.

SIL Level	Risk Reduction Factor (RRF)	Probability of Failure on Demand (PFD)
SIL 1	10 - 100	10^{-1} to 10^{-2}
SIL 2	100 - 1k	10^{-2} to 10^{-3}
SIL 3	1k - 10k	10^{-3} to 10^{-4}
SIL 4	10k - 100k	10^{-4} to 10^{-5}

Table 2.1: Safety integrity level - target failure measures for a safety function operating in low demand mode of operation.

Achieving a specific SIL level requires meeting several design requirements aimed at ensuring system robustness and reliability:

- **Fault Tolerance**: Higher SIL levels require increased fault tolerance, often

achieved by system redundancy (e.g., dual or triple channels).

- **Failure Rate Calculations**: SIL assessment involves evaluating component failure rates, determining their effects on the safety function, and ensuring they meet reliability targets.

- **Testing Frequency and Diagnostics**: High SIL levels mandate more frequent testing, diagnostics, and maintenance to detect and rectify faults before they compromise safety.

For example, a SIL 1 system might require single-channel architecture with basic diagnostics and lower testing frequency, by contrast, a SIL 3 system might require dual-channel redundancy, sophisticated diagnostics, and frequent testing to detect and repair potential failures.

2.2.1.2 PL (Performance Levels)

Performance Level (PL) is defined in ISO 13849-1 [21]. It is another metric used to assess the reliability of safety-related parts of control systems (SRP/CS). PL evaluates the safety of control systems in terms of their resistance to dangerous failures. Unlike SIL, which focuses on Probability of Failure on Demand (PFD), PL is determined through an analysis that includes the Severity (S) of potential harm, the Frequency and Duration of Exposure (F) to the hazard, and the Possibility of Avoidance (P). These three factors (S, F, and P) help define the necessary performance level for a system.

PL levels are categorized from PL a (lowest performance level) to PL e (highest performance level). Each level represents a probability of dangerous failure per hour (PFH), which can be translated into the system's required level of reliability and robustness.

PL	Probability of Dangerous Failure per Hour (PFH)	Risk Reduction Factor (RRF)
PL a	$\geq 10^{-5}$ to $< 10^{-4}$	Low
PL b	$\geq 10^{-6}$ to $< 10^{-5}$	Moderate
PL c	$\geq 3 \cdot 10^{-7}$ to $< 10^{-6}$	Significant
PL d	$\geq 10^{-7}$ to $< 3 \cdot 10^{-7}$	High
PL e	$\geq 10^{-8}$ to $< 10^{-7}$	Very High

Table 2.2: Performance level - target failure measures for a safety function operating in low demand mode of operation.

Higher PL levels represent lower probabilities of failure, with PL e being the highest level of safety and reliability. The required PL level is determined by evaluating the potential risk using the three main factors (S, F, and P). Once the required PL is defined, designers use ISO 13849-1 to select or design safety components and architectures that meet or exceed this level. Key factors in achieving the appropriate PL include:

- **Diagnostic Coverage (DC)**: PL requires a certain diagnostic coverage, i.e., a system's ability to detect failures before they become dangerous.
- **Category of Safety Control System**: The standard defines categories (B, 1, 2, 3, 4) that relate to structural design requirements, from basic designs (B) to highly redundant architectures (4).
- **Mean Time to Dangerous Failure (MTTFd)**: It quantifies the average time before a dangerous failure occurs and is an important factor in achieving higher PL levels.

For example, a PL b system could use basic single-channel architecture with moderate diagnostics, while a PL e system might require multi-channel redundancy, high diagnostic coverage, and frequent inspections.

2.2.1.3 Comparing SIL and PL

While both SIL and PL address functional safety, their differences make them suitable for various applications:

- **Scope**: SIL is often used for complex programmable control systems, while PL is common in discrete machinery and simpler safety-related systems.
- **Assessment Factors**: SIL focuses on PFD, while PL considers severity, frequency, and avoidance potential, making PL a practical choice for machinery-specific applications.
- **Complexity**: SIL involves more stringent analysis, typically requiring tools for failure probability calculations; PL is generally simpler and includes both electronic and mechanical systems.

In practice, EN 62061 (for SIL) and ISO 13849-1 (for PL) can be used alongside each other to ensure compliance with Directive 2006/42/EC, allowing manufacturers of machinery like Mobile Elevating Work Platforms (MEWPs) to meet stringent safety requirements for control systems and operations.

2.3 Speed sensing

In mobile robotics, the accurate measurement of speed is fundamental for achieving robust control and navigation. Speed information allows the robot to maintain stable movement, adapt to environmental changes, and execute precise maneuvers, which are essential for both autonomous and teleoperated mobile platforms. Speed sensors, therefore, serve as a critical component of the sensory architecture, enabling the robot to gauge its own motion dynamics with precision and reliability. Among the various types of speed sensors, encoders stand out due to their precision, adaptability, and the wealth of information they can provide on both speed and position, making them indispensable in a wide array of robotic applications.

2.3.1 Encoders

Encoders work by converting mechanical motion into electrical signals, which can then be interpreted to determine both the speed and position of a moving component, such as a wheel or motor shaft. This information is important in closed-loop control systems, where the robot continually adjusts its actions based on real-time feedback from its environment. The value of encoders in robotics lies in their ability to deliver precise feedback even at high speeds, thus enabling smooth motion control, efficient trajectory following, and consistent response in complex or dynamic environments.

Encoders are often mounted directly on the drive motors or wheels, facilitating real-time measurement of linear or angular velocity. This information enables the robot to adjust its velocity and orientation dynamically, providing a higher degree of control and autonomy.

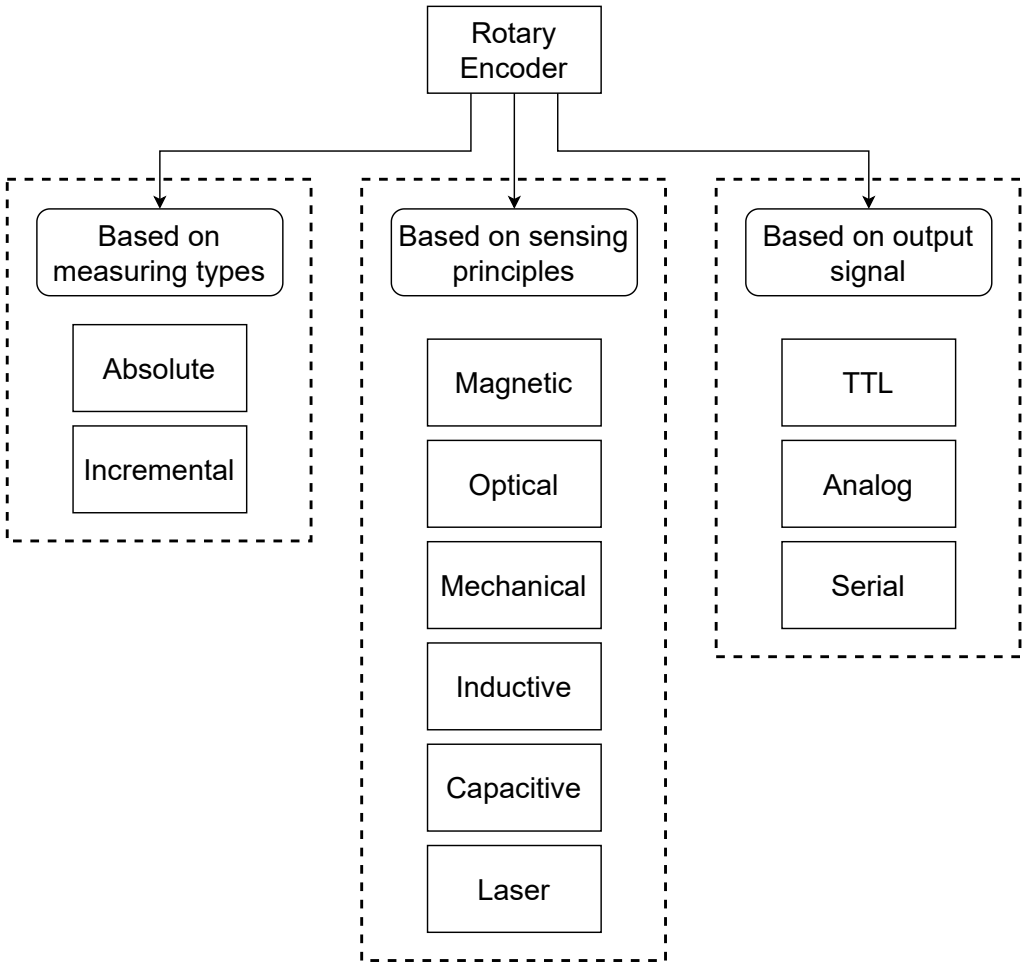

Figure 2.1: Encoders classification scheme

Encoders come in a variety of types and configurations (see Fig. 2.1), each tailored to specific applications and environmental conditions. Key performance attributes such as resolution, accuracy, and robustness are crucial considerations when selecting an encoder, as they directly affect the robot's overall performance. High-resolution encoders allow for finer speed and position measurement, enhancing the robot's ability to make precise adjustments. However, factors like environmental durability, signal noise immunity, and size constraints also play a role in determining the suitability of a specific encoder type for a particular application.

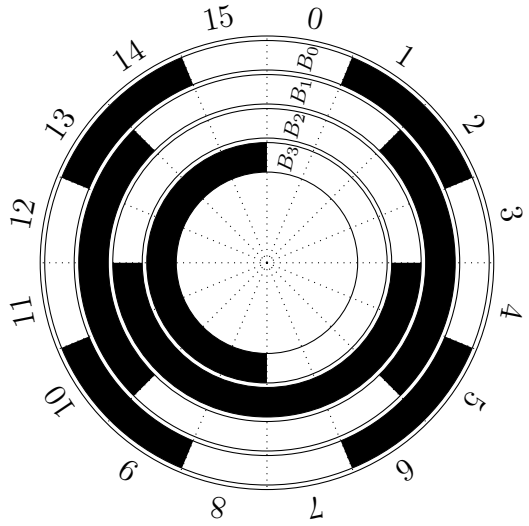

Figure 2.2: Absolute encoder wheel

Encoders are commonly classified by their sensing mechanisms, with optical encoders and magnetic encoders being the most prevalent in mobile robotics (see Fig. 2.1). These encoders leverage different physical principles to convert rotational or linear motion into electrical signals, each type offering unique advantages and limitations in terms of resolution, robustness, and environmental suitability. The choice between encoder types is often influenced by the operational requirements of the robot, such as the need for high accuracy in indoor environments or the demand for durability in outdoor or dusty settings. For our system, we have chosen differential incremental encoders with TTL output as the speed sensors, furthermore, to exploit the backlash phenomenon, we considered the encoder mounted after the gearbox. Unlike absolute encoders, incremental encoders do not provide information about the absolute position of the rotor to which they are attached. However, this is not a problem since our control objective is velocity rather than absolute position.

An incremental encoder consists of a disk divided into N slots as depicted in

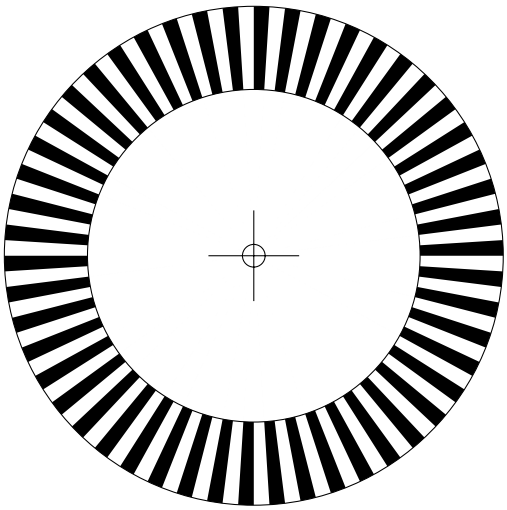

Figure 2.3: Incremental encoder wheel

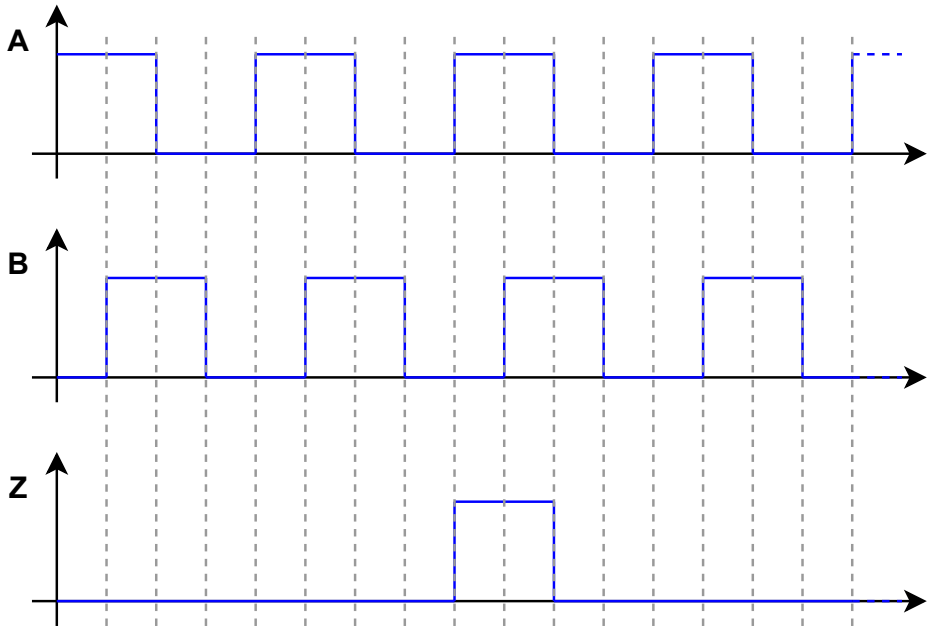

Figure 2.4: Square wave incremental encoder

Fig. 2.3. A light-emitting diode (LED) and a photodiode are positioned on the sides of the encoder disk. As the disk rotates, interrupting the light emitted by the LED, the sensor transmits a signal containing information about the shaft's

positional change. These impulses form a square wave as in Fig. 2.4. The longer the period of this wave, the lower the rotational speed of the motor shaft. Conversely, a higher frequency of impulses represents a higher speed. This square wave signal is transmitted as a dual signal on two channels (denoted ch=A,B) that are phase-shifted by $\pi/2$ radians, allowing us to determine the direction of rotation. A third channel, denoted as Z, provides an absolute reference position of the encoder shaft.

While the instantaneous speed cannot be determined with encoders, the average speed within an observation time ΔT can be calculated by counting the number of rising and falling edges using the formula:

$$N_{ch} = \sum_{k=1}^{K} \frac{|V_{ch}(k) - V_{ch}(k-1)|}{V_{enc}} \tag{2.1}$$

Normalized with respect to the encoder's supply voltage V_{enc}, this formula allows to determine the number of edges within a time ΔT, where K represents the number of samples acquired during the observation. This computation is performed for both channels A and B. In addition, V_{ch} denotes the voltage of the channel.

The rotational speed of the encoder is expressed as:

$$\omega_{ch} = \frac{2\pi}{4PPR} \frac{N_{ch}}{\Delta T} \tag{2.2}$$

Where, Pulses Per Revolution (PPR) represents the encoder resolution, quantified as the number of impulses per revolution of the encoder disk. The denominator of the rotational speed of the encoder is multiplied by 4 times the PPR because quadrature decoding doubles the count for each state change of both channel A and B, resulting in 4 times the count for each pulse or period.

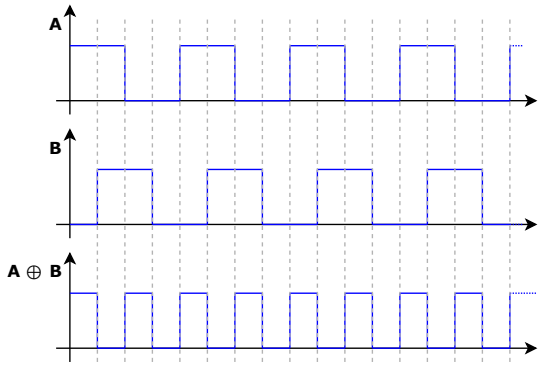

Figure 2.5: Representation of signals originating from channels A and B of the encoder. The third plot depicts the outcome derived from the exclusive OR (XOR) operation applied to these channels ($A \oplus B$).

2.4 Differential Drive vehicles

Differential drive vehicles are a category of mobile platforms primarily employed in industrial settings that enhance maneuverability when compared with the conventional Ackermann kinematics. The fundamental mechanics of differential drive vehicles involve two (or more) independently driven wheels. Two or more motors allow the rotational speed of each wheel to be independently set, enabling the execution of precise maneuvers, including forward and backward movement, pivot turns, and complex curves (see Fig. 2.6). A forklift is a prime example of a differential drive vehicle used in warehouses and factories, where its capacity for intricate movements makes it invaluable for lifting and transporting heavy loads.

Central to the distinctive capabilities of differential drive vehicles is their steering mechanism, which is realized by the possibility of setting different speeds for each wheel using one electric motor for each wheel. For instance, to initiate a right turn, the left wheel accelerates while the right one decelerates, causing the vehicle to pivot around its central axis. This results in exceptional agility and accuracy, making these vehicles suitable for applications demanding precise control

in constrained spaces and uneven terrains.

A common practice in differential drive vehicles to obtain precise speed control and navigation is to use encoders on each wheel to allow closed-loop control of the speed. These encoders provide real-time feedback on the wheel rotation speed and position. However, it is of the utmost importance to consider that these encoders are susceptible to failures, such as signal loss or sensor malfunctions, potentially leading to a risky uncontrolled vehicle.

Therefore, despite their advantages, these vehicles present specific technical challenges in terms of safety and reliability. Ensuring the dependable operation of these vehicles, even in the face of sensor and encoder issues, asks for advanced fault-tolerant control models and safety measures which is what this paper focuses on.

The kinematics of a differential drive vehicle can be modeled as follow [22]:

$$v_c = \frac{r(\omega_R + \omega_L)}{2} \quad \omega_c = \frac{r(\omega_R - \omega_L)}{L} \tag{2.3}$$

where: v_c is the linear speed of the vehicle, ω_c is the rotational speed of the vehicle, r is the radius of the wheels, L is the distance between the center of the wheels and ω_i with $i = L, R$ is the rotational speed of the wheels, respectively left and right.

2.4.1 A Safe Control system

In order to operate a safe feedback control of the left and right wheels in a differential drive vehicle, we propose the four-state machine depicted in Fig. 2.7.

The starting state is the *IDLE* state, wherein the machine is expected to remain stationary while the controller awaits commands from the operator.

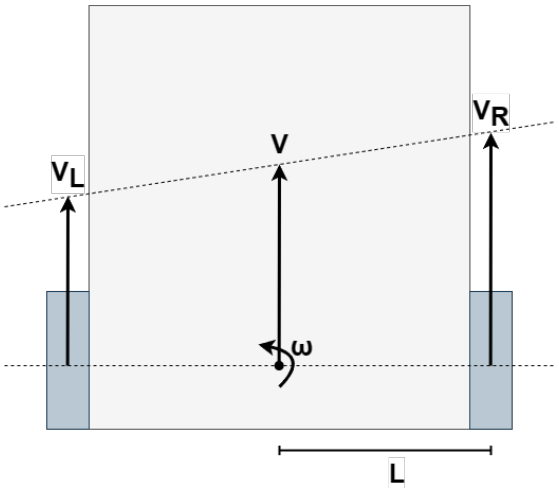

Figure 2.6: Scheme of differential drive vehicle kinematics

Following the operator's issuance of commands, the machine changes to the *ARMED* state, where a meticulous test on the encoder's functionality must be executed. If the test is successful, then the machine is allowed to enter the *MOVING* state; otherwise, the machine enters the *FAIL* state.

The *MOVING* state is characterized by the controller's nominal operation, involving also real-time analysis of sensor data to identify potential failure conditions. The controller stays in this state until the operator issues further commands or until a system failure is detected.

Upon the detection of a failure, the *FAIL* state is entered by the controller, which is a secure mode. This failure may happen during the encoder test at the startup time or when the machine operates in the *MOVING* state. When in the *FAIL* state, all outputs to motor drivers are restrained, thereby cutting power and stopping any movement. Upon the operator's release of commands, the controller gracefully reverts to the *IDLE* state, providing a mechanism for resetting false positive detections.

In case the detected failure is authentic, upon the operator's subsequent command issuance in the *ARMED* state, the startup test failure ensues, compelling

the controller to return to the *FAIL* state. If the fault detected is authentic, the next command issued by the operator in the *ARMED* state will cause the startup test to fail, forcing the controller to return to the *FAIL* state.

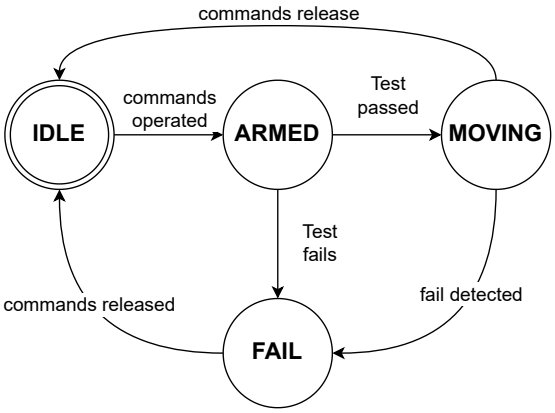

Figure 2.7: The proposed state machine to operate a safe controller

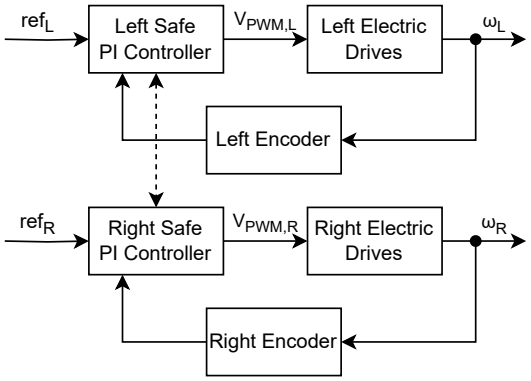

Figure 2.8: Block diagram of a safe control system for a differential drive vehicle

2.5 Fault Scenarios

In industrial environments, vehicles are exposed to a variety of fault scenarios that can impact their safety and operational efficiency. These scenarios can be categorized by both the severity of their impact and the frequency with which

they occur, providing a structured approach to fault management and response prioritization.

Faults with high severity pose immediate risks, potentially leading to unsafe conditions or significant disruptions if not addressed instantly. Critical scenarios, such as brake failure, loss of steering control, or major sensor faults that compromise obstacle detection, require the vehicle's control system to initiate rapid interventions. Emergency protocols may involve immediately stopping the vehicle, shifting to backup systems, or alerting nearby personnel to prevent accidents or equipment damage.

In contrast, low-severity faults may not create an immediate hazard but can still degrade the system's performance over time. These faults, including slight sensor deviations, intermittent communication issues, or minor wear on components, are typically less urgent but still require monitoring and timely intervention to avoid cumulative effects. By adjusting operational parameters or scheduling regular maintenance, these minor faults can be managed before they escalate into critical issues.

The frequency of fault occurrences further shapes the response strategy, influencing maintenance schedules and resource allocation. High-frequency faults, even those of low severity, can disrupt workflow and reduce system reliability if they recur too often, as these scenarios necessitate frequent inspections, calibrations, or part replacements. Conversely, low-frequency faults may appear less disruptive in the short term but could indicate underlying issues that warrant attention to prevent future critical failures.

Categorizing fault scenarios by severity and frequency enables a balanced approach to both safety and efficiency. High-severity, high-frequency faults might demand immediate corrective action or design modifications, while low-severity,

low-frequency issues can be managed with routine maintenance without impacting productivity. This structured framework for understanding and addressing fault scenarios optimizes the operational reliability and safety of industrial vehicles in demanding environments.

2.5.1 Sensor Faults

Sensor faults, particularly those affecting incremental encoders, are critical concerns in mobile robotics as they directly impact the accuracy of speed and position measurements. Incremental encoders are vital components that translate mechanical motion into electrical signals, providing feedback necessary for precise control of the robot's movement. Faults in these sensors can lead to significant errors in navigation, control algorithms, and overall system performance. The primary categories of encoder faults are detailed below:

- **Encoder Detachment ($\Delta E_{1,L}$ and $\Delta E_{1,R}$):** This fault occurs when the encoder becomes physically detached or loose from the motor shaft to which it is connected. Causes may include mechanical vibrations, improper installation, or gradual loosening over time due to wear and tear. When an encoder detaches, it may shift along the shaft axis or completely disconnect, leading to either zero output or unstable and erratic signals. This severe fault is relatively rare but poses a high risk to system functionality, as it disrupts the feedback loop essential for accurate speed and position control. The immediate consequence is a loss of reliable data, which can cause the robot to misinterpret its speed, potentially resulting in unsafe behavior or collisions. Preventive measures include secure mounting techniques, regular inspections, and the use of locking mechanisms to maintain encoder alignment.

- **Encoder Stuck ($\Delta E_{2,L}$ and $\Delta E_{2,R}$):** An encoder may become stuck due to mechanical obstructions such as debris, corrosion, or internal component failures like bearing seizure. Electrically, issues like short circuits, power supply anomalies, or signal interference can also cause the encoder to output a constant value regardless of actual motion. When the encoder is stuck, it provides unchanging speed measurements, misleading the control system into believing the robot is stationary or moving at a constant speed. This fault occurs with higher frequency compared to encoder detachment and is considered critical because it can lead to improper control commands, inefficient operation, or unsafe conditions. For instance, the robot might fail to decelerate when approaching an obstacle. Addressing this fault involves implementing real-time monitoring systems that detect anomalies in encoder output patterns and trigger fault-handling protocols, such as switching to alternative sensors or entering a safe operational mode.

- **Generic Encoder Failure ($\Delta E_{3,L}$ and $\Delta E_{3,R}$):** This category encompasses a broad range of electrical and mechanical issues that result in random or nonsensical outputs from the encoder. Potential causes include intermittent wiring connections due to frayed cables, degradation of optical components in optical encoders (e.g., LED burnout or photodetector failure), and magnetic interference affecting magnetic encoders. Environmental factors like extreme temperatures, moisture ingress, or exposure to contaminants can exacerbate these issues. Although such failures are infrequent, their severity is high because unpredictable encoder signals can lead to erratic robot behavior, making it difficult for the control system to make accurate decisions. Mitigation strategies involve robust sensor design with environmental protections, redundancy through multiple sensors, and ad-

vanced fault detection algorithms that can filter out erroneous data and rely on predictive models or alternative feedback mechanisms during failure conditions.

The impact of sensor faults extends beyond immediate control loops to higher-level functions such as path planning, localization, and mapping. Inaccurate sensor data can compromise the robot's ability to build and interpret maps of its environment, affecting long-term autonomy and mission success. Therefore, integrating fault-tolerant designs, regular maintenance schedules, and comprehensive diagnostics is essential for ensuring reliable sensor performance in mobile robotic systems.

2.5.2 Actuator Faults

Actuator faults in electric drive systems are critical to both the performance and safety of mobile robots. Electric actuators, typically DC motors controlled by power drivers, are responsible for executing movement commands issued by the control system. Faults in these components can result in loss of mobility, unintended movements, or even hazardous situations. The main actuator faults considered are elaborated below:

- **Motor Detachment (ΔM_L and ΔM_R):** This fault occurs when the electrical connection between the motor and its power driver is interrupted. Possible causes include loose wiring due to vibrations, connector failures from mechanical stress, or faults within the driver circuitry such as blown fuses or damaged components. The immediate effect is a temporary loss of motor function on the affected side, which can lead to asymmetrical thrust and unintended deviations from the desired trajectory. While the severity

is considered low because the robot may still retain partial functionality, this fault can compromise mission objectives and requires timely detection and remediation. Preventive measures involve secure electrical connections, strain relief for cables, and the use of connectors designed to withstand the operational environment's mechanical stresses.

- **Driver Short-Circuit (ΔSC_L and ΔSC_R):** A short-circuit within the motor driver, particularly between two channels of the H-bridge circuit, can cause the motor terminals to be directly connected to the power supply without the regulating influence of control signals. This condition can lead to uncontrolled motor behavior, such as the motor running at maximum speed uncontrollably or experiencing a sudden stop due to the lack of proper voltage regulation. The fault is severe due to the risks of overheating, excessive current draw, potential damage to the motor windings, and safety hazards including fires or electrical shocks. Although rare, the critical nature of this fault necessitates immediate shutdown procedures within the control system, as well as hardware protections like fuses, circuit breakers, and current-limiting resistors. Regular diagnostic checks and the use of high-quality driver components with built-in fault protections can reduce the likelihood of such occurrences.

- **Driver Breakdown (ΔDB_L and ΔDB_R):** A driver breakdown involves the failure of components within the H-bridge motor driver, such as transistors (e.g., MOSFETs or IGBTs) or diodes, leading to unintended electrical pathways that can expose the motor directly to the supply voltage or ground. This fault can cause abrupt and unregulated motor movements, stalls, or oscillations, posing significant safety risks and potentially damaging the motor

and other electrical components due to voltage spikes or reverse currents. The severity of this fault is high, and while its occurrence is infrequent, it underscores the importance of using drivers with robust designs that include protections against overvoltage, overcurrent, and thermal overload. Implementing fault-detection circuits that monitor driver health and incorporating software routines that can isolate faulty drivers can enhance system resilience. Additionally, designing the system to fail safely, where the robot enters a controlled stop upon detecting such faults, can prevent accidents and equipment damage.

Managing actuator faults effectively requires a combination of proactive and reactive strategies. Proactively, the use of high-reliability components, protective circuit designs, and adherence to proper installation and maintenance procedures can minimize the occurrence of faults. Reactively, the implementation of real-time monitoring systems that track motor currents, voltages, temperatures, and driver statuses enables the early detection of anomalies. Software algorithms can analyze this data to predict potential failures and initiate appropriate responses, such as reducing motor load, switching to redundant systems, or alerting operators. By integrating these approaches, mobile robots can maintain high levels of performance and safety even in the face of actuator faults.

2.5.3 Safety requirements

The safety requirements are outlined in this subsection, and the fault detection time T_D, for the corresponding failures, is defined as multiple of the control sampling time T_s.

- **False detections**: The number of false positives must be kept as low as

Fault	Severity	Occurency	Detection time
$\Delta E_{1,L}$,$\Delta E_{1,R}$	Very high	Low	$T_D < 50T_s$
$\Delta E_{2,L}$,$\Delta E_{2,R}$	Very high	Medium	$T_D < 150T_s$
$\Delta E_{3,L}$,$\Delta E_{3,R}$	High	Low	$T_D < 150T_s$
ΔM_L,ΔM_R	Low	Low	$T_D < 50T_s$
ΔSC_L,ΔSC_R	Very high	Very Low	$T_D < 50T_s$
ΔDB_L,ΔDB_R	Low	Low	$T_D < 50T_s$

Table 2.3: Considered faults

possible to avoid the vehicles becoming unusable. We have required that the minimum time between consecutive false positives should be less than 360,000 samples (approximately 1 hour). In the event of a false positive detection, the controller will push the machine into the *FAIL* state, easily re-set by the operator by releasing the commands to activate the machine and run the startup test again. The detection will be either confirmed (correct detection) or refuted (false positive).

- **Missed detections**: All faults should be detected.

- **Safe state behavior**: An essential feature is to define a policy regarding the controller's behavior in the event of fault detection. In case of a failure, the controller must promptly bring the speed of both motors to zero. This is achieved by reducing to zero the power supplied to both motors and, if available on the vehicle, engaging the brakes.

2.5.4 Controller

The vehicle controller operates within a state machine, as described in the section 2.1. The controller is implemented with a sampling frequency of 100Hz, and it starts in the *IDLE* state.

The condition for moving from *IDLE* to *ARMED* is that $r_L \neq 0 \vee r_R \neq 0$. It

is evident that, as with industrial vehicles, safety regulations require measures to prevent accidental activation of the vehicle.

In the *ARMED* state, the system performs a pre-departure check of the encoder and motor functionality. This test leverages the backlash within the gearbox [23]. Regulations dictate that the vehicle cannot be activated without the explicit intention of the operator. Typically, a test involving motor activation would violate this requirement. However, by utilizing the backlash within the gearbox, it is possible to conduct a micro-actuation on the motor shaft that is not transmitted to the wheel.

Algorithm 1 Algorithm of the start test

$direction \leftarrow 0$ ▷ $0 = CCW, 1 = CW$
`test procedure:`
$actuateMotorsForTest(direction)$
while test is running **do** recordEncodersData()
end while
$N_A, N_B \leftarrow countPulseCh(A, B)$ ▷ Eq. 2.1
$\omega_A, \omega_B \leftarrow computeSpeedCh(A, B)$ ▷ Eq. 2.2
$AB \leftarrow computeXORChannels()$
$\omega_{AB} \leftarrow computeSpeedCh(AB)$
$C_1 \leftarrow N_A - (N_A \mod 2) == N_B - (N_B \mod 2)$
$C_2 \leftarrow \omega_A == \omega_B$
$C_3 \leftarrow \omega_{AB} == 2\omega_A$
if $C_1 \wedge C_2 \wedge C_3$ **then** ▷ Test passed
    $testPassed \leftarrow 1$
else if direction is 0 **then**
    $direction \leftarrow 1$ ▷ Retry changing direction
    `goto test procedure`
else
    $testPassed \leftarrow 0$
end if

The motor is driven with a very low pulse width modulation (PWM) value, insufficient to move the vehicle but sufficient to rotate the motor shaft freely within the backlash zone. If, during this operation, waveform edges aligned with

the model are detected, the test is considered successful.

If no waveform edge is detected at the encoder input, the test is repeated by activating the motor in the opposite direction, since the gear attached to the motor shaft might be in contact with the gear attached to the driving wheel. If this operation detects waveform edges that match the model, the test is considered successful. Otherwise, if none are detected, the test is deemed definitively failed, indicating a possible disconnection of the encoder or a broken motor driver. The controller prevents the vehicle from moving, signals the malfunction, and pushes the machine into the *FAIL* state.

Specifically, we check equation (2.4) to make sure that the number of edges detected on both channels, if even, is the same. In the case of an odd number of edges on one channel, we subtract 1.

$$N_A - (N_A \mod 2) = N_B - (N_B \mod 2) \tag{2.4}$$

After confirming this, we make sure that the detected speed is the same for both channels of the encoder. If this speed corresponds, it will be the measured rotational velocity: $\omega_A = \omega_B \triangleq \omega$.

Confirming that the number of edges and the velocity are the same on both channels is not enough to ensure safety. One of the two channels (or both) could show anomalous behavior and manage to overcome the first two checks. Therefore, we proceed to perform an XOR operation between the two channels ($A \oplus B$) and calculate the measured velocity ($\omega_{A \oplus B}$). By performing the XOR operation, if the signals are correct, we double the signal frequency. If there are no problems with the encoder, we must verify that: $\omega_{A \oplus B} = 2\omega$.

If a higher-than-expected speed is detected during the test, the test will stop

and fail. This could indicate a short circuit in the motor driver. The controller prevents the vehicle from moving, signals the failure, and puts the machine in the *FAIL* state.

If the test is successful, the controller enters the *MOVING* state. In this state, the PI controller operates normally, adjusting the motor speed to the required setpoint. During this state, the motor current consumption and speed measurements from the encoder are monitored. If the data deviates from the model, a fault is detected, and the machine enters the *FAIL* state. Real-time analysis of the sensor data verifies that the difference between the sensor estimate and the actual value is always below a certain threshold.

Finally, in the *FAIL* state, motor activation is inhibited, preventing any movement of the machine.

The PI controllers for both motors, although operating independently, communicate with each other when they detect a fault. When one controller detects a failure, it communicates this information to the other controller, and both simultaneously enter the *FAIL* state.

2.6 Experiments and results

This section describes the experimental validation of the proposed fault detection system. To assess the vehicle dynamics, it is assumed that the operator follows the trajectory shown in Figure 2.9. Moving along this trajectory, it is possible to experiment with right and left movements. Moreover, the experiment has focused on the forward movement along this trajectory since the backward movement along the same trajectory does not add any further insight.

In the experimental setup, we built a prototype vehicle shown in Figure 2.10,

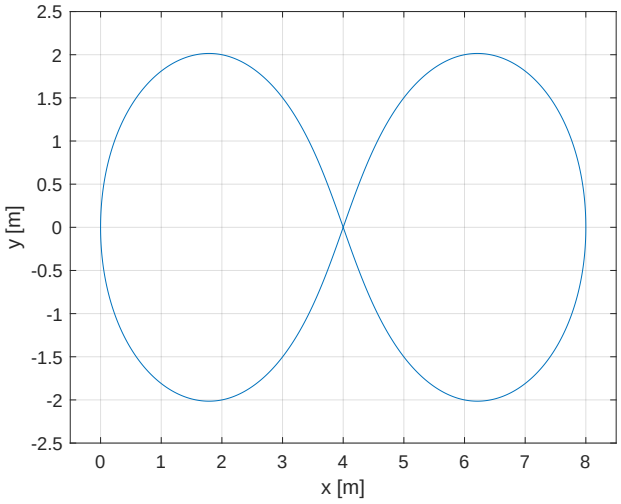

Figure 2.9: Vehicle trajectory considered in the experiment

where we have employed 24V DC motors. Hall effect differential encoders were mounted on the motor shafts. We utilized a custom control board with a Microchip microprocessor and a custom power driver capable of working with motors up to 2kW. The experiments lasts for 5 minutes.

Failures should be detected and managed according to the requirements specified in Section 3.3. To execute the validation, random failures have been injected during the experiment. These experiments were repeated 100 times to exhaustively validate the safety system.

Figure 2.11 shows an excerpt of an experiment. It shows the speed set point (blue solid line), the speed measured by the encoder (Dashed red curve) and the actual speed (dotted black curve). At the beginning of the experiment, there is a short transient period after which the motor stabilizes at the reference speed. At t=23s, a fault is injected that is detected a few milliseconds later, leading the motor into a quiescent safe state.

Figure 2.10: Prototype vehicle.

2.7 Concluding Remarks

This chapter has focuses on developing fault-tolerant control systems for industrial vehicles, particularly those with differential drive kinematics. These vehicles present unique safety challenges due to potential malfunctions, so reliable failure detection and mitigation are crucial.

While there has been substantial research on fault-tolerant control systems, much of it focus on motor and torque control, which less attention is given to rotary speed sensor failures.

Experimental results validate the proposed system, showing that it effectively detects failures and transitions the motor to a safe state. This ensures that the vehicle can continue operating safely even in the event of a sensor fault. The approach meets safety standards like the EU Directive 2006/42/EC and IEC61508, making it a valuable contribution to improving industrial vehicle safety.

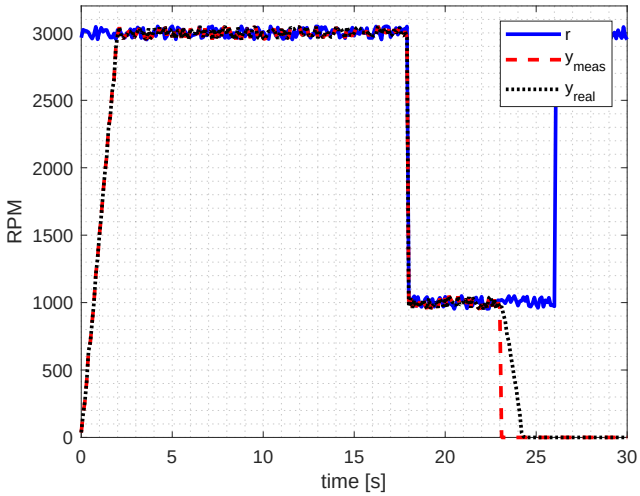

Figure 2.11: Excerpt of a experiment with fault injection.

Chapter 3

Decentralized Control of UAV Swarms

3.1 Introduction

Unmanned Aerial Vehicles (UAVs), especially multi-rotor drones, are increasingly being used for a wide range of tasks, such as photogrammetry [24], remote sensing [25], search and rescue [26], surveillance [27], and firefighting [28]. These drones are equipped with various sensors like RGB and RGB-D cameras, LiDARs, and radars, which allow them to capture data that can be processed locally or transmitted in real-time to a Ground Control Station (GCS). A UAV swarm involves multiple drones working together on a shared task, offering advantages like increased coverage, efficiency, and resilience. Swarms are particularly valuable for complex tasks such as collaborative mapping, multi-sensor data fusion, and distributed sensing. However, to function effectively, each drone in the swarm needs to have autonomous capabilities, such as path planning and obstacle avoidance, as well as coordinated control strategies to ensure smooth collaboration.

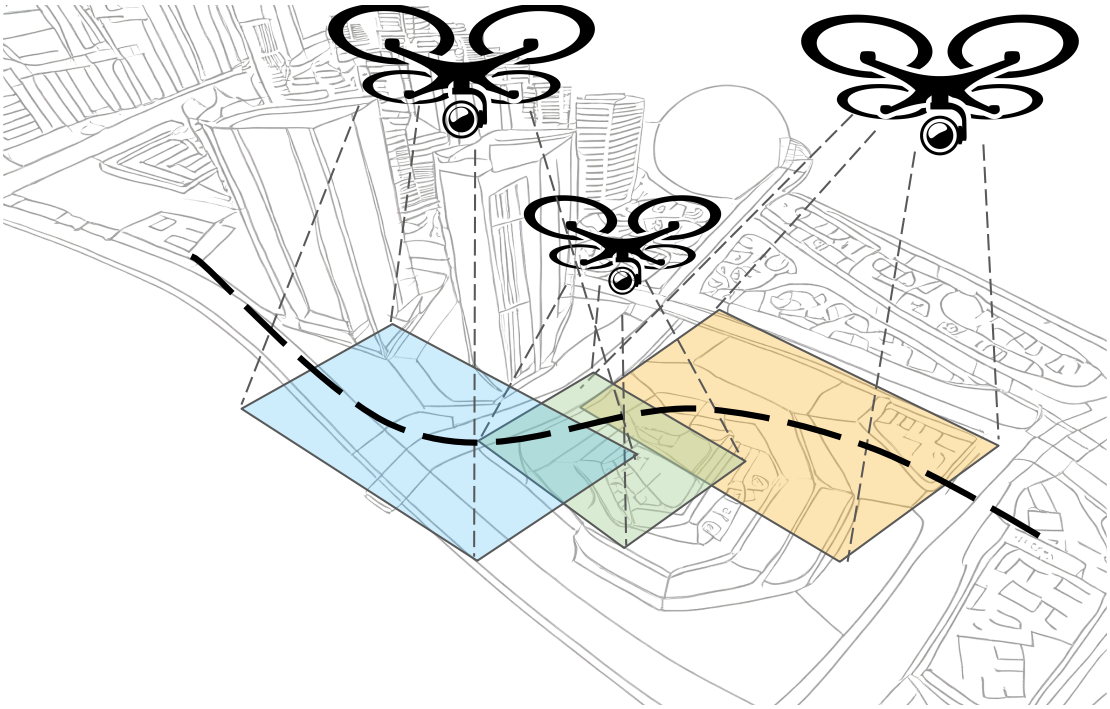

Figure 3.1: A swarm of drones autonomously patrols a given area of interest while sending videos to a ground station

Control strategies for UAV swarms are often based on methods developed in the field of Multi-Agent Systems (MAS), including swarm intelligence [29], formation control [30], consensus algorithms [31], and decentralized control [32].

Tasks like surveillance and patrolling often require real-time video streaming from the drones to the GCS. UAV swarms are especially useful in these scenarios because they can improve mission efficiency, increase coverage, and enhance situational awareness. The task we focus on in this study involves a swarm of drones tracking a path, capturing video with onboard cameras, and sending the footage back to the GCS, typically via a public communication network.

We aim to achieve two main goals: First, we want to maximize the area covered by the drones along their path, ensuring that the field of view of each drone overlaps with its neighbors. This overlap allows for dynamic video stitching at the GCS, giving a complete view of the area. Second, we address the impact

of fluctuating network bandwidth on video quality. Low bandwidth reduces the quality of the videos sent to the GCS, which can affect the accuracy of computer vision tasks. To mitigate this, we propose dynamically adjusting the drones' altitude in response to bandwidth fluctuations. Lowering the drones' altitude increases the pixel density of the video, improving its quality. However, this also creates challenges in maintaining the desired overlap between drones, making video stitching more difficult.

To tackle these issues, we propose using a partially distributed Nonlinear Model Predictive Control (NMPC) framework. NMPC is well-suited to handle problems with constraints, such as limitations in drone control, energy consumption, and coverage. It has been successfully applied to tasks like collision avoidance, navigation, and formation control.

In our approach, each drone is subject to realistic constraints, such as limited control inputs and energy consumption, which ensures both efficiency and practicality in real-world applications.

3.2 Related Work

UAV surveillance, area coverage, and video streaming are rapidly evolving fields that have attracted attention from a variety of research communities, including Artificial Intelligence (AI), Graph Theory, and Multimedia Systems. In this section, we provide a brief overview of existing research and identify the gaps our work aims to address.

The Swarm-based UAV (SUAV) algorithm proposed in [33] is designed for detecting mobile objects during natural disasters, such as floods, using a swarm of drones. The algorithm focuses on energy-efficient tracking while considering

visual quality metrics like SSIM and VQM based on drone speed. It also aims to maintain multi-hop communication links between the drones and the ground to ensure better image quality. However, this work does not account for the impact of fluctuating network bandwidth on video transmission.

In [34], the authors tackle the problem of cooperative exploration with multiple UAVs, ensuring that they maintain line-of-sight (LOS) communication with the ground through a relay UAV. While the study considers collision avoidance and feasible flight paths, it doesn't explore how time-varying network bandwidth might affect communication and performance, which is a key concern in our work.

The paper by [35] explores how to maximize coverage while minimizing resource consumption, specifically by considering the power usage of drones through a Particle Swarm Optimization (PSO) approach. However, this solution does not use a collaborative approach or direct communication between drones, as our work does. Instead, it assumes that drones scan separate areas without interaction.

In [36], the authors discuss UAV-based surveillance with cameras to cover large areas. While they consider network-related issues, such as packet loss, they do not take into account how changing network bandwidth affects video quality.

The study in [37] focuses on the communication between UAVs and the ground station, specifically considering bandwidth for coverage path planning. It employs a centroidal Voronoi diagram to assign coverage areas to each UAV without direct communication. In contrast, our work uses a multi-agent system framework that not only ensures efficient coverage but also incorporates bandwidth-aware motion control, enabling drones to communicate directly and avoid redundant movements and collisions.

Another related approach is found in [38], where decentralized monitoring using drones is achieved through distributed optimal control algorithms that min-

imize information loss. This approach allows for adding or removing drones from the team, but it lacks an active collision avoidance system. Unlike this work, our approach focuses on improving video quality by adjusting the drones' movements based on network bandwidth, ensuring both efficient coverage and high-quality video capture. While our work assumes a predefined path, we recognize that path planning methods like those in [39] could be adapted for real-time path generation.

3.3 System design

This section outlines the system design used to achieve the goals introduced in the previous section. First, we define the notation and concepts that will be used throughout the chapter (Section 3.3.1). Then, the scenario and the proposed framework are presented in Sections 3.3.2 and 3.3.3, respectively.

3.3.1 Preliminaries

Let $\|\cdot\|_p$ represent the p-norm of a vector, and consider a *directed graph* (digraph) $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of nodes (indexed as $i = 1, \ldots, N$), and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. An edge (v_i, v_j) indicates the flow of information from node i to node j. The set of neighbors of a node v_i is denoted by $\mathcal{N}_i = \{v_j \in \mathcal{V} : (v_j, v_i) \in \mathcal{E}\}$, meaning all nodes that have directed edges towards node v_i. A directed graph is said to be strongly connected if there is a directed path between every pair of nodes, i.e., for any nodes i and j, there is a path from i to j and vice versa.

The adjacency matrix A captures the structure of the graph, with entries a_{ij} such that $a_{ij} = 1$ if there is an edge from v_i to v_j (i.e., $(i, j) \in \mathcal{E}$), and $a_{ij} = 0$ otherwise. The in-degree matrix $D = D(G) = (d_{ij})$ is defined such

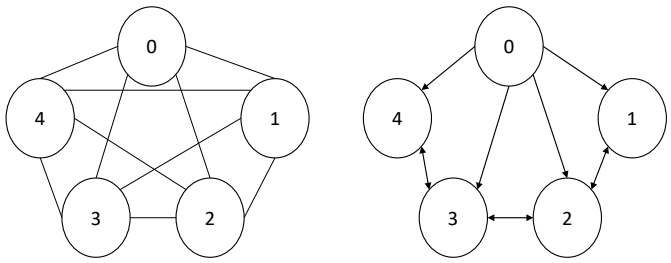

Figure 3.2: a) Communication graph b) Actuation graph

that $d_{ij} = \text{in-deg}(v_i)$ if $i = j$ (the in-degree of node v_i), and $d_{ij} = 0$ otherwise. The in-degree of node v_i, $\text{in-deg}(v_i)$, represents the number of edges directed towards node v_i. The Laplacian matrix is then given by $L = D - A$. If G is strongly connected, the eigenvalues of the matrix $-L$ satisfy the condition $-\lambda_{N-1} \leq -\lambda_{N-2} \leq \cdots < -\lambda_0 \leq 0$, where $\lambda_0 = 0$ and the rank of L is $N - 1$.

Formation Control is a well-established technique in multi-agent systems used to maintain a particular shape or distance between agents. In this paper, we apply this concept to ensure that agents maintain a safe minimum distance from each other, while the shape of the formation is allowed to change over time. The goal is to achieve a *flexible* formation, which ensures that safety distances are maintained without rigidly enforcing a fixed formation shape. At the same time, it is important that the agents stay close enough to each other to allow their on-board cameras to scan overlapping areas for tasks such as stitching and object detection.

3.3.2 Scenario definition

In this subsection, we describe the scenario and the control goals that we aim to achieve. Imagine a swarm of drones equipped with cameras, like the one shown in Figure 3.1. These drones are tasked with tracking a predefined path and monitoring an area (such as a field, a town, or other environments) for various purposes, including firefighting, border patrolling, agricultural monitoring, surveillance, object detection, and more. Each drone's camera is pointed directly downward, capturing a square area whose size depends on the drone's altitude. The video feeds from these cameras are sent to a Ground Control Station (GCS) through a network connection, such as a 5G link. The GCS receives the video streams from all the drones in real-time and stitches the frames together to create a composite video that covers the entire area beneath the drones.

To stitch the video frames together effectively, the algorithm requires some overlap between the captured areas, meaning the videos need to share common ground coverage, and the quality of the video needs to be high [40, 27]. The first step in the stitching process is detecting common features in the video frames coming from different drones. For this to work well, the videos must be high-quality, or else the stitching algorithms, which rely on identifying feature points in the images, might fail. Additionally, since some overlap is required between the videos, the drones need to stay as close to each other as possible without colliding.

From a practical perspective, each drone has an Internet connection to communicate with other drones and with the GCS to send the captured video. We assume that each drone can exchange information with all the other drones, forming a complete directed graph. In real-world scenarios, the available network bandwidth depends on the drone's location in the area and the number of users sharing the

network. As a result, bandwidth is variable and not known in advance, and it is treated in this work as a disturbance that can be measured. Techniques for estimating bandwidth—both passive and active—are available and can be used for this purpose [41, 42].

Because the bandwidth can vary, the quality of the video each drone sends also changes. When the bandwidth is low, it is preferable for the drone to lower its altitude to get closer to the ground, allowing it to capture more detailed frames that can help with the video stitching process at the ground station. On the other hand, when the bandwidth is high, the drone can increase its altitude, expanding the area it covers while maintaining good video quality. A key parameter for measuring the quality of the captured image is the *Ground Sample Distance* (GSD), which describes the size of a pixel on the ground and is an important concept in UAV-based photogrammetry [43].

At the same time, the drone swarm needs to maintain a wide coverage area. Ideally, the drones should fly as far apart as possible, but without leaving gaps in the areas they are scanning. To achieve this, the drones would need to fly at their highest possible altitude, allowing them to capture the largest possible area.

In this scenario, each drone has two conflicting goals: i) ensure a good amount of overlap between the videos and maintain high visual quality in each video, and ii) increase the coverage of the region of interest. To achieve the first goal, a drone would fly at a lower altitude, which results in a smaller GSD and better video quality. However, increasing the altitude improves coverage because the camera can capture a larger area of the ground.

To balance both goals, we adopt the following approach: when the bandwidth is high, the drone can fly higher to increase coverage, while still maintaining good resolution and visual quality. However, when the bandwidth is low, the

drone descends, as previously mentioned. To achieve this, a mapping between the altitude and available bandwidth must be defined so that the video quality is only minimally affected.

Additionally, to monitor the area effectively, one drone is chosen as the leader to follow a predefined path. The other drones in the swarm are required to follow the leader while maintaining a safe distance from both the leader and the other drones. It is important to note that the altitude of each drone will change as they move along their path, based on the current network bandwidth.

3.3.3 Problem definition

In this section, we introduce the Nonlinear Model Predictive Control (NMPC) that is executed on each drone. Specifically, we have designed a two-state algorithm to prevent detachments—situations where there is no overlap between the videos captured by the drones. Avoiding this condition is crucial for the video stitching process to work correctly. For each state, we define a specific cost function for the NMPC. It is also worth noting that the drones communicate with each other using the same Internet connection they use to send video streams to the Ground Control Station (GCS). The bandwidth is assumed to be constant in the vertical (z) direction and over time.

Drone Dynamics and Cost Functions

The dynamics of each drone are modeled in a simple way as discrete-time integrators. This simplification helps describe how each drone's state evolves over time.

$$s_i(k+1) = s_i(k) + T_s u_i(k), \quad i = 1, \ldots, N \tag{3.1}$$

Here, N is the total number of drones, $s_i = (x_i, y_i, z_i)^T$ is the position vector of drone i, T_s is the sampling time, and $u_i = (u_{x,i}, u_{y,i}, u_{z,i})$ represents the control input for drone i.

Additionally, the 3D orientation of the drones is not considered in this model. This is because we assume that the cameras are mounted on gimbals, which are set to always point straight down at the ground with the same yaw angle for all drones.

Now, we are ready to introduce the proposed NMPC designed to solve the problem. The general structure of the optimization problem is as follows:

$$\begin{aligned}
\underset{u_i}{\text{Minimize}} \quad & \mathcal{J}_i && (3.2)\\
\text{subject to} \quad & s_i(k+1) = s_i(k) + T_s u_i(k) && (3.3)\\
& h_{\min} \leq h_i \leq h_{\max} && (3.4)\\
& \|X_i(t) - X_j(t)\|_2 \geq 2R_{\text{safe}} && (3.5)\\
& |u_i| \leq u_{\max} && (3.6)\\
& |\Delta u_i| \leq \Delta u_{\max} && (3.7)
\end{aligned}$$

In these equations, $X_i = (x_i, y_i)^T$ is the 2D position of drone i, h_i is the altitude of the drone, and Δu_i is the change in control input between two consecutive time steps. Equation (3.4) ensures that the drone's altitude stays within a specific range, which is set for safety reasons and can be considered a *mission parameter*. Equation (3.5) guarantees that each drone maintains a minimum distance of $2R_{\text{safe}}$ from other drones to avoid collisions. Equations (3.6) and (3.7) are constraints on the control input and its rate of change, which help ensure that the control actions remain realistic.

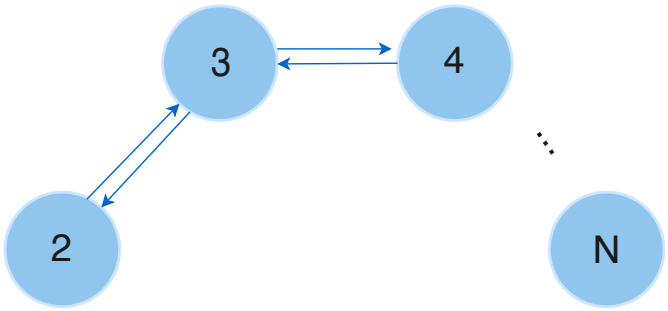

Figure 3.3: Graph used for the computation of J_f

One key point to note is that the NMPC for the leader drone, denoted by $i = 1$, does not include the collision constraint (3.5). This is because the leader's main job is to follow a predefined path, and it's up to the other drones to ensure they maintain a safe distance from the leader.

Finally, we describe the two-state Track-Recovery algorithm, where the cost function $\mathcal{J}_i$ is defined depending on the specific state and the drone in question.

Track-Recovery Algorithm

The swarm can operate in one of two possible states: the *Tracking State* and the *Recovery State*. In the Tracking state (T), the risk of detachment is minimal, as there is enough overlap between the areas covered by the cameras. On the other hand, in the Recovery state (S), there is a significant risk of detachment, so the cost function within the NMPC framework needs to be adjusted for each agent. During the Recovery state, the leader drone remains stationary, and the system transitions back to the Tracking state once the required overlap is reestablished among the agents.

State T - Tracking

In the Tracking state, the leader follows a desired path, and the other agents follow the leader, ensuring that the swarm achieves optimal coverage and video overlap. Each agent also adjusts its altitude dynamically based on the available network bandwidth to maintain the necessary video quality.

For this state, we define the cost function for the leader, $\mathcal{J}_{1,T}$, for problem (3.2)-(3.7) excluding the collision constraint (3.5), as described earlier. To avoid redundancy, the following equations are written generically with index i. However, for the leader's cost function, $i = 1$.

$$\mathcal{J}_{1,T} = w_p J_p + w_h J_h + w_u J_u + w_{\Delta u} J_{\Delta u}, \tag{3.8}$$

$$J_p = \sum_{k=1}^{N_p} \left\| X_i(t+k|t) - \bar{X}(t+k) \right\|_\infty \tag{3.9}$$

$$J_h = \sum_{k=1}^{N_p} \left\| h_i(t+k|t) - \bar{h}_i(t+k) \right\|_2 \tag{3.10}$$

$$J_u = \sum_{k=1}^{N_c} \left\| u_i(t+k-1) \right\|_2 \tag{3.11}$$

$$J_{\Delta u} = \sum_{k=1}^{N_c} \left\| \Delta u_i(t+k-1) \right\|_2 \tag{3.12}$$

Here, J_p represents the cost of path tracking, where $X_1(t)$ is the 2D position of the leader at time t, N_p is the prediction horizon, and $\bar{X}(t)$ is the reference path at time t. J_h is related to altitude tracking, where $h_1(t)$ is the leader's altitude

and $\bar{h}_1(t)$ is the reference altitude. J_u and $J_{\Delta u}$ are terms that limit control effort and changes in control input. The control horizon is denoted by N_c.

For the other agents, $\mathcal{J}_{i,T}$ for $i = 2, \ldots, N$ is given by:

$$\mathcal{J}_{i,T} = w_l J_l + w_h J_h + w_u J_u + w_{\Delta u} J_{\Delta u}, \tag{3.13}$$

$$J_l = \sum_{k=1}^{N_p} \left(\|X_i(t+k|t) - X_1(t+k)\|_\infty - r(R_i + R_1) \right)^2 \tag{3.14}$$

Here, J_l encourages agent i to follow the leader's path.

State S - Recovery

In some cases, the swarm risks losing its formation, which is critical for maintaining video overlap among agents. This can lead to *detachments*, where at least one agent's video feed does not overlap with any other agent's video. Since each agent knows the positions of all other agents, they can detect this risk and switch to the *Recovery* state (S). In this state, the leader halts and waits for the other agents to reassemble. Once the agents have properly aligned and adjusted their altitudes, the system transitions back to the Tracking state (T).

The leader's cost function during the Recovery state ensures it does not follow the path and instead waits for the others to catch up. Meanwhile, the leader adjusts its altitude according to the reference altitude, while penalizing control input and its variations. The cost function is defined as:

$$\mathcal{J}_{1,S} = w_h J_h + w_u J_u + w_{\Delta u} J_{\Delta u} \tag{3.15}$$

For the other agents in the Recovery state, the cost function does not penalize

control input or its variation, allowing them to use maximum control effort to recover quickly while still respecting input and variation constraints.

Furthermore, agents are required to stay close to one another. To avoid issues with altitude tracking or leader following, we use a communication graph with fewer edges in the Recovery state. Specifically, we consider a topology without the leader, as shown in Figure 3.3, where each agent communicates with one or two other agents. For this simplified graph, we define $\mathcal{N}_i = \{j \in \mathcal{V} : (j, i) \in \mathcal{E}\}$ to represent the neighbors of agent i. The term J_f in the cost function ensures that the agents stay together and maintain some overlap in their scanning areas through the term $r(R_i + R_j)$, where R_i and R_j are the coverage radii for agents i and j, respectively. J_f is defined as:

$$J_f = \sum_{j \in \mathcal{N}_i} \sum_{k=1}^{N_p} \left(\|X_i(t+k|t) - X_j(t+k)\|_\infty - r(R_i + R_j) \right)^2 \tag{3.16}$$

From simple geometric principles, we know that the relationship between R_i and h_i is given by $R_i = h_i \tan(\phi/2)$, where ϕ is the angular aperture of the video camera [44]. The parameter r controls the desired overlap: setting r close to 1 results in minimal overlap, while lowering r brings the agents closer together. Although the communication graph is a complete directed graph (since every agent can communicate with any other agent via the Internet), the subgraph used for J_f is reduced to avoid issues with altitude tracking and redundant communication. This approach of using different graph topologies for specific tasks has been discussed in the literature, such as in [45], where three topologies—Sensing Topology, Communication Topology, and Actuation Topology—are introduced, each designed for different goals.

The cost function for the other agents, $i = 2, \ldots, N$, in the Recovery state is then:

$$\mathcal{J}_{i,S} = w_l J_l + w_h J_h + w_f J_f \tag{3.17}$$

Lastly, the weights $w_p, w_l, w_f, w_h, w_u, w_{\Delta u}$ need to be carefully tuned to achieve the best performance. This tuning can be done empirically or using optimization tools.

Depending on the agent and its state, the cost function in (3.2) can be $\mathcal{J}_{1,T}$, $\mathcal{J}_{i,T}$, $\mathcal{J}_{1,S}$, or $\mathcal{J}_{i,S}$ for $i = 2, \ldots, N$.

Correlation Between Bandwidth and Altitude

To determine $\bar{h}_i$ in (3.10) for each agent, we use the following function to relate the estimated available bandwidth b to $\bar{h}_i$, as explained in Section 3.3.2:

$$\bar{h}_i(b) = \begin{cases} h_{\min} & \text{if } b < b_{\min} \\ S \cdot b + h_{\max} - S \cdot b_{\max} & \text{if } b_{\min} \leq b \leq b_{\max} \\ h_{\max} & \text{if } b > b_{\max} \end{cases} \tag{3.18}$$

where S is defined as $S = \frac{h_{\max} - h_{\min}}{b_{\max} - b_{\min}}$.

This function could be replaced by a different model that better captures the relationship between altitude, visual quality, and available bandwidth. Identifying such a model is outside the scope of this study and could be a topic for future work.

3.4 Simulations

In this section, we present the results of simulations that validate the effectiveness of the proposed approach. We consider several scenarios with different numbers of agents, specifically $N \in \{3, 5, 6\}$, and for three predefined paths: a sine-like path, a semicircular path, and a generic path designed to simulate a road. The key parameters, including the weights in the cost functions, are summarized in Table 3.1.

Table 3.1: Values of the weights in the cost functions of the NMPCs

	w_l	w_p	w_f	w_h	w_u	$w_{\Delta u}$
$\mathcal{J}_{1,T}$	1.5	-	-	5	0.001	0.001
$\mathcal{J}_{i,T}$	-	0.5	-	1.5	0.001	0.001
$\mathcal{J}_{1,S}$	-	-	-	0.5	0.001	0.001
$\mathcal{J}_{i,S}$	1.5	-	0.3	1.5	-	-

For the NMPC, the parameters were set to $r = 0.4$, $N_p = 5$, $N_c = 2$, $R_{\text{safe}} = 2$, $h_{\min} = 4$, and $h_{\max} = 10$. The upper and lower bounds for $\Delta u_x, \Delta u_y, \Delta u_z$ were set to 0.1 for all agents. To ensure the swarm maintained its formation, the leader's velocity was intentionally set slower than that of the follower agents. Specifically, the leader's velocity bounds were $u_{x,\max} = 0.7$, $u_{y,\max} = 0.7$, while the followers were allowed to go up to $u_{x,\max} = 1$ and $u_{y,\max} = 1$. For altitude control, $u_{z,\max}$ was set to 0.1 for all agents to avoid abrupt altitude changes.

The network bandwidth was modeled using a uniform random distribution, varying between 500 kbit/s and 5000 kbit/s, to simulate realistic conditions. While no specific dataset is available for drone channel measurements based on geographic position, this distribution serves as a reasonable approximation for the simulations.

3.4.1 Overall Results

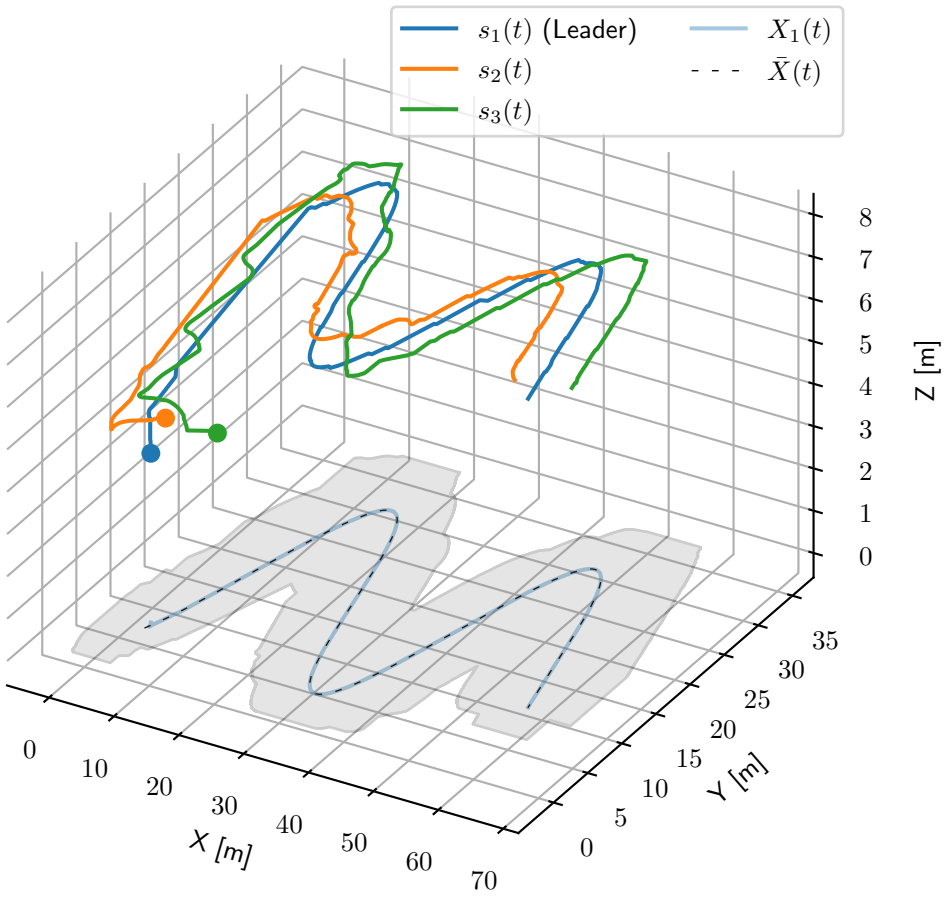

Figure 3.4: Drones' trajectories and area covered for $N = 3$

First, we examine the case of $N = 3$, with the leader following a sine-like path. The drones' trajectories and the area they cover are shown in Figure 3.4. Here, the agents start from their initial positions (marked with round markers) and attempt to follow the leader while maintaining formation. The leader's reference path is shown by the black dashed line, and the leader's trajectory (projected onto the x-y plane) is depicted in light blue, demonstrating good tracking. The gray region represents the area covered by the swarm, and the z-axis shows the altitude, which varies along the path based on available bandwidth.

Figure 3.5 illustrates the case with $N = 6$ agents tracking the same sine-

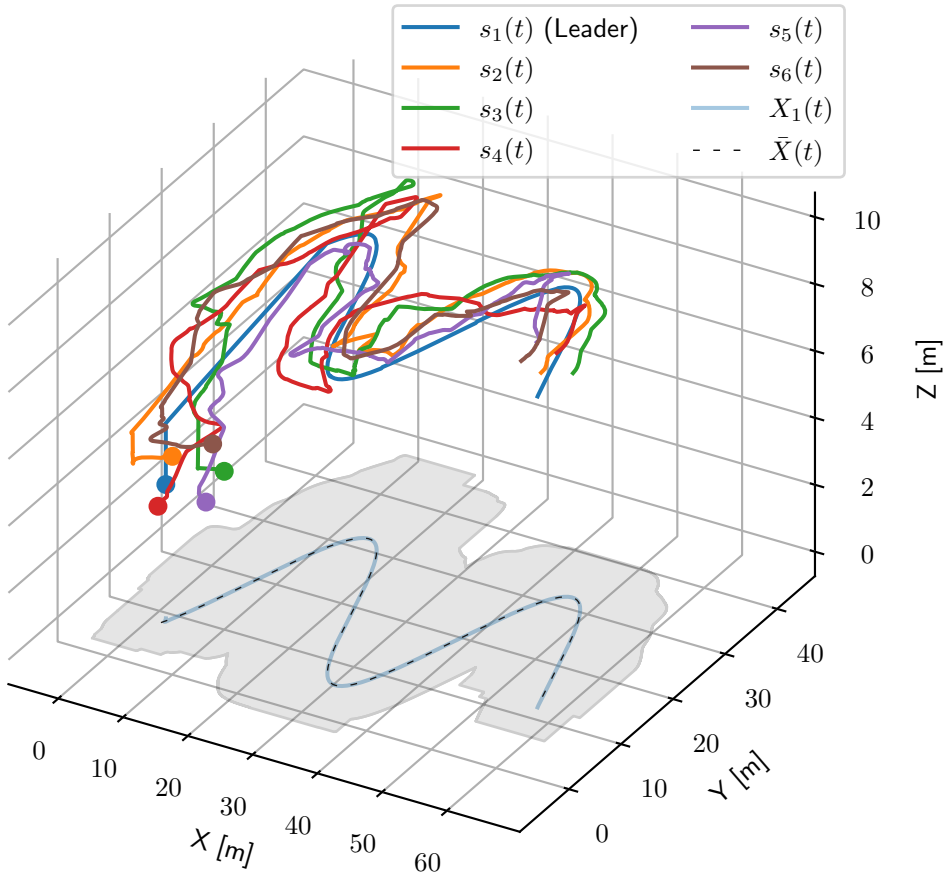

Figure 3.5: Drones' trajectories and area covered for $N = 6$

like path. As expected, the agents stay well-coordinated, ensuring overlap and preventing collisions. The larger swarm also results in significantly more area being covered compared to the $N = 3$ case. Similar results were observed for $N = 5$ agents and for the other paths considered.

Figure 3.6 shows the altitude tracking error for the $N = 6$ case. Each agent starts at the same altitude, and after a brief transient period, all agents quickly converge to their reference altitudes. The altitude for each agent is dependent on its position and the available bandwidth at each step. Similar results were observed for the other cases.

As expected, increasing the number of agents leads to a larger total scanned area. Table 3.2 shows the normalized scanned areas for $N = 3, 5, 6$. As the

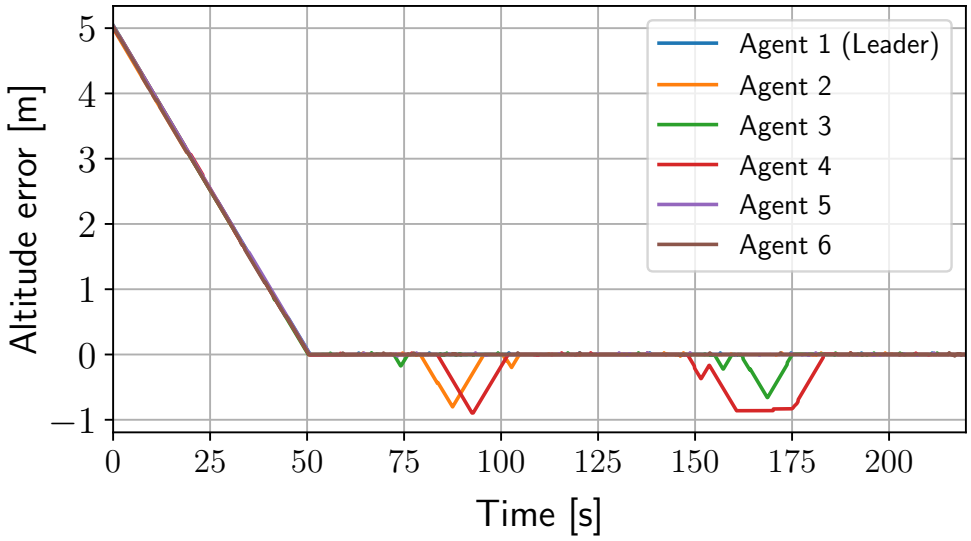

Figure 3.6: Altitude tracking error for $N = 6$

number of agents increases, the area covered grows accordingly, with about a 30% increase in area when moving from $N = 3$ to $N = 6$, across all three paths.

3.4.2 In-depth Analysis

Next, we perform a sensitivity analysis with $N = 5$ agents tracking the sine-like path. We vary the leader's maximum velocity, $u_{x,\max}$ and $u_{y,\max}$, within the set $\{0.7, 0.8, 0.95\}$ to understand the impact on system performance. We also evaluate the time the swarm spends in the *Recovery* state, during which the leader pauses to allow the followers to catch up. The higher the time spent in Recovery, the less efficient the system becomes.

We define an *efficiency index* η as follows:

$$\eta = 1 - \frac{T_S}{T_{\text{tot}}} \tag{3.19}$$

where T_S is the time spent in Recovery, and T_{tot} is the total time taken for the mission. A higher η indicates better performance, as it means less time spent in

Recovery.

Our results show that when the leader's velocity is set to 0.7, the efficiency is $\eta = 0.9$. Increasing the leader's velocity to 0.8 results in a decrease in efficiency to 0.86, and further increasing to 0.95 leads to an efficiency of 0.82. This confirms that higher velocities for the leader lead to more time spent in the Recovery state, reducing the overall efficiency.

	$N = 3$	$N = 5$	$N = 6$
Sine	1.0	1.3	1.35
Semicircle	1.0	1.3	1.38
Generic	1.0	1.31	1.37

Table 3.2: Normalized scanned area with respect to $N = 3$

3.4.3 Comparison with Existing Work

Finally, we compare our approach with two existing strategies from the literature: the *Independent* strategy, where each agent follows its own path independently, and the *Centralized* strategy, where all agents follow the same reference path, with the leader moving at a constant speed and the followers maintaining relative distances.

The results show that our approach outperforms both the Independent and Centralized strategies in terms of coverage and coordination. The Independent strategy struggles with inefficient coverage and the potential for collisions, while the Centralized strategy is better at maintaining formation but more prone to detachment issues when the leader moves too quickly. Our approach provides better area coverage while keeping the swarm coordinated, avoiding detachment, and ensuring efficient task completion.

3.5 Concluding Remarks

A decentralized NMPC-based control framework for a swarm of drones has been proposed to monitor a designated area through onboard cameras. These cameras capture videos that are transmitted in real-time to a ground station. To maximize video quality, the method considers both available bandwidth and the altitude of each drone.

The proposed framework utilizes leader-follower multi-agent system formation control to maintain close proximity among drones, preventing collisions and ensuring sufficient overlap in the captured videos for stitching at the Ground Control Station. The leader drone follows a specified path, with the other agents adjusting their positions accordingly. Each agent applies NMPC to optimize a cost function that facilitates all these objectives.

Specifically, the algorithm incorporates two operational states, namely *Tracking* and *Recovery*. These states are designed to minimize potential gaps in the area covered by the drones. In each state, the agents are assigned distinct cost functions tailored to the objectives of that state. Results demonstrate the effectiveness of the proposed approach, which ensures video overlap for stitching, precise path and altitude tracking, and adequate area coverage.

Part II

Video Streaming Systems

Chapter 4

Introduction

Video streaming is a fundamental component in mobile robotics, since it enables the transmission of visual data that are critical for tasks such as remote monitoring, teleoperation, and autonomous navigation. Advancements in wireless communication and embedded computing on mobile robotic platforms made possible to implement real-time video streaming, even under challenging operational conditions. Video streaming for mobile robotics presents unique challenges, such as ensuring low latency, maintaining bandwidth efficiency, and optimizing energy consumption.

The demand for high-quality, real-time video streaming is essential for a wide range of robotic applications, including search and rescue, industrial automation, and environmental surveillance [6]. These applications require robust and efficient video streaming systems capable of supporting real-time data-driven decision-making processes. Although progress in adaptive streaming protocols, data compression, and low-latency networking has improved video streaming capabilities, further advancements are necessary to address the specific requirements of mobile robotics.

One of the major challenges in streaming video for mobile robots involves balancing the trade-offs between video quality, latency, and bandwidth usage. High-definition video demands greater bandwidth and processing power, which may be unavailable in remote or bandwidth-limited environments. As a result, adaptive streaming techniques and optimized data compression algorithms have become critical in managing these resources effectively.

Moreover, latency is a crucial factor in mobile robotics, as delays in video transmission can negatively impact the robot's performance, particularly in time-sensitive applications.

Emerging technologies such as edge computing and advanced wireless communication protocols (e.g., 5G and 6G) offer promising solutions to reduce latency, yet more research is needed to adapt these technologies to the constraints of mobile robotics.

In this chapter, we explore recent advancements in video streaming technologies aimed at enhancing the performance and reliability of mobile robotic systems. We review adaptive streaming protocols, examine the role of data compression in optimizing bandwidth usage, and discuss the impact of cutting-edge wireless communication technologies on latency. Our goal is to provide a comprehensive overview of these improvements and their implications for mobile robotic applications.

4.1 The Adaptive Video Streaming Model

A video v is encoded into different representations or *levels* $l \in \mathcal{L}$, where each representation corresponds to a nominal encoding bitrate level l_i.

Each video level is divided into K segments of fixed duration τ. Video segments are identified by an index $s \in \mathcal{S}$, where $\mathcal{S} = \{1, 2, \ldots, K\}$ is the set of video segment indices. Let $c : \mathcal{L} \to \mathbb{R}$ denote a non-decreasing function that maps a selected bitrate video level l_s with the video quality perceived by the user while playing the s-th segment.

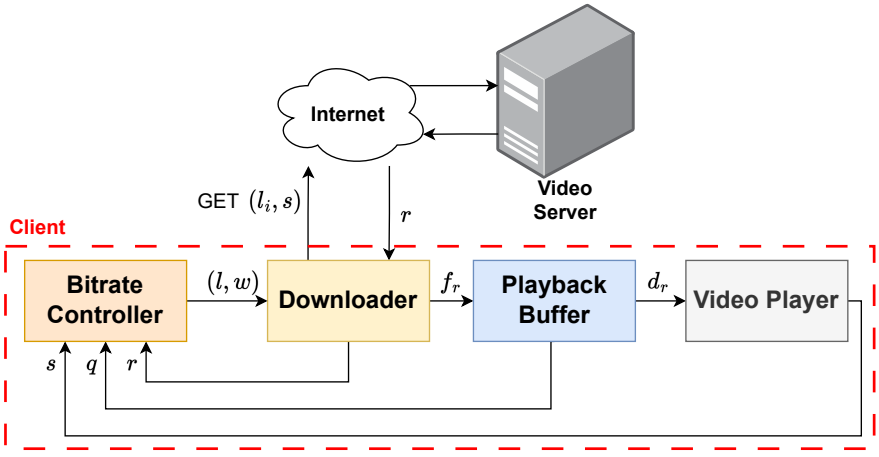

Figure 4.1: Abstract model of an adaptive video streaming system.

To formulate the mathematical model for a video streaming system, we have drawn upon the analysis carried out in [46] and depicted in Figure 4.1. When a streaming session is initiated, the controller, whose goal is to optimize the Quality of Experience (QoE) perceived by the user, instructs the downloader to request and download video segments encoded at the proper quality level that are stored in the playback buffer to be displayed.

The control law is computed upon completion of each segment download and is followed by the download of the next chunk at the bitrate level $l_s \in \mathcal{L}$ computed

by the controller's output $u_s \in \mathcal{L}$.

The downloader fetches video segments from the server at the bitrate determined by the controller.

More precisely, the download of segment s starts at time t_s, and ends after the download time t_{D_s}, i.e., the time interval between t_s and the completion of the download of segment s. After that, the downloader waits for w_s seconds before requesting the next segment $s+1$. This waiting time depends on the time required to compute the level of the next chunk to be downloaded. The downloader stops if the queue length in the playback buffer exceeds the threshold Q_H and waits until it is below the threshold before resuming the download. This approach is useful when the video is paused by the user, as it prevents unnecessary data network resources consumption.

Hence, the download of segment $s + 1$ occurs at:

$$t_{s+1} = t_s + t_{D_s} + w_s \tag{4.1}$$

Let B_s be the average bandwidth measured during the download of the s-th chunk and $C_s(u_s)$ the size of the s-th chunk downloaded at bitrate $u_s \in \mathcal{L}$, then t_{Ds} can be expressed as:

$$t_{Ds} = \frac{C_s(u_s)}{B_s} \tag{4.2}$$

The *video player* is responsible for draining the playout buffer, decoding, and rendering the video on the user's screen. Notice that the player's state at time t, denoted as $\alpha(t)$, can be equal to 1 when the video is playing and the playout buffer is being drained, or 0 when the video is paused and no video is drained from the playout buffer. The player is in the latter state when either the buffer is empty

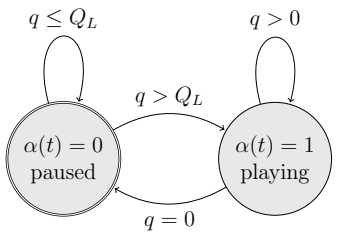

Figure 4.2: Depiction of player state as finite state machine.

or at the beginning of a video streaming session. While in this state, the player remains paused until the playback buffer is filled up to a threshold Q_L. The state of the player can be modelled as the finite state machine shown in Figure 4.2.

The video playback rate P_R is the rate at which the video content is drained from the buffer and played on the user's screen. Although such a rate can be slightly adjusted for synchronization purposes in the context of live video streaming events [47], it is in general considered constant and equal to 1, which means that 1 second of video is played in 1 second.

Hence, the player's drain rate can be expressed as follows:

$$d_r(t) = P_R \cdot \alpha(t) \tag{4.3}$$

The client's playback buffer, which stores the video content, has a length $q(t)$ measured in seconds. The downloader fills the playback buffer with an impulse of amplitude τ seconds when segment s is downloaded at t_{s+1}. Then, the player drains the buffer through the continuous-time process (4.3).

Thus, the queue length resulting after downloading the s-th segment can be expressed as:

$$q(t_{s+1}) = q(t_s) + \tau - \int_{t_s}^{t_{s+1}} d_r(\xi)\, d\xi \tag{4.4}$$

Before starting a streaming session, the server sends to the client all the nec-

essary information, such as the set $\mathcal{L}$, the video clip duration T, and the chunk length τ, through a manifest file in DASH standard or Playlist file in HLS standard.

4.2 Quality of Experience in video streaming

Quality of Experience (QoE) is defined as *"the degree of delight or annoyance of the user of an application or service. It results from the fulfilment of his or her expectations with respect to the utility and/or enjoyment of the application or service in the light of the user's personality and current state"* [48]. In the context of communication services, QoE is influenced by a complex interplay of factors, which include not only the technical performance of the network but also the content, the device, the specific application, the user expectations, and the specific context in which the service is used [49].

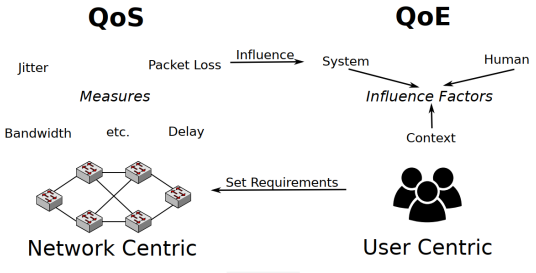

Figure 4.3: Conceptual difference of QoS and QoE [1]

Quality of Service (QoS) and QoE are both essential concepts in assessing the performance of communication services; however, they represent fundamentally different perspectives. While QoS is a network-centric metric that quantifies the technical performance of a service (such as latency, bandwidth, and packet loss), QoE is user-centric, aiming at capturing the subjective perception of service quality from the user's perspective (Fig. 4.3).

QoS is primarily concerned with measurable network performance indicators and provides a quantifiable view of service quality. In contrast, QoE relies on a broader, multidimensional perceptual space encompassing:

- **System Factors:** These include technical attributes, such as QoS metrics (e.g., packet loss, latency), network protocols, and device-specific parameters. While these factors can impact user satisfaction, they are not sufficient to capture the full spectrum of user experience.
- **Human Factors:** Subjective elements such as mood, personality traits, expectations, and previous experiences significantly impact QoE. For instance, a user's mood or their expectations from a service may alter their perception of quality independently of the measured QoS.
- **Contextual Factors:** Context, including the user's physical location, the nature of the task, and even cost considerations, can greatly influence QoE. For example, a user may tolerate lower quality when streaming a video for leisure but may demand higher quality in professional or critical applications.

These factors do not operate in isolation but often interact, complicating the mapping between QoS and QoE. For instance, users may rate their experience differently with the same service depending on external factors such as environment or expectations, indicating that QoE depends on a broader set of influence factors that extend beyond measurable QoS metrics.

4.2.1 QoE metrics

In order to grasp users' perception of video quality in streaming, the industry and researchers depend on particular Quality of Experience (QoE) metrics, referred

to as Key Performance Indicators (KPIs). These key performance indicators for quality of experience in adaptive video streaming gather important elements of user perception, offering understanding of service quality from the user's point of view. Overall, we can say that several factors could be taken into account when trying to estimate the user's QoE[50, 51].

- **Average Video Quality**: The bitrate determines the quality of the video, usually measured in kilobits per second (kbps). In streaming applications where quality is adjusted based on network conditions, the average bitrate indicates the overall quality experienced throughout the session, considering all displayed bitrates and their durations.

- **Average quality switch**: Counting the number of times the video quality switches during the session measures how well the stream adjusts to network fluctuations and reliability.

- **Buffering Ratio**: Video playback pauses temporarily when there is no more content to display. This break, known as buffering, can greatly affect the watching experience. The buffering ratio shows how much time the video player is paused for buffering compared to the total session time.

- **Buffering Frequency**: The rate at which buffering interruptions happen in a streaming session, calculated by dividing pauses by session duration.

- **Startup Delay**: This measure calculates the duration, in seconds, from when a user initiates a video to when it starts playing. Another related idea, rebuffering delay, is when there is a pause in video playback before it resumes.

A metric that is widely adopted in the literature to estimate the QoE is defined as [2]:

$$QoE_{lin}{}_1^K = \underbrace{\sum_{k=1}^{K} c(u_k)}_{\text{Quality reward}} - \lambda \underbrace{\sum_{k=1}^{K-1} |c(u_{k+1}) - c(u_k)|}_{\text{Smoothness penalty}} - \mu \underbrace{\sum_{k=1}^{K} r_s}_{\text{Rebuffering penalty}} - \mu_s \underbrace{T_s}_{\text{Startup delay}} \tag{4.5}$$

where λ, μ, μ_s are non-negative weighting parameters. The *Quality reward* term denotes the total reward in terms of the visual quality of each level. The *Smoothness penalty* term penalizes video level switch. The *Rebuffering penalty* term negatively affects the QoE and is the sum of all the rebuffering durations r_s for any chunk $s \in \mathcal{S}$. Finally, the *Startup delay* is the time the player requires before starting to reproduce the video.

Name	**Quality reward** $c(u_k)$	**Rebuffering penalty weight** μ
QoE_{lin}	$c(u)$	4.3
QoE_{log}	$log(c(u)/c(u_{min}))$	2.66
QoE_{hd}	$0.3 \to 1, 0.75 \to 2, 1.2 \to 3, 1.85 \to 12, 2.85 \to 15, 4.3 \to 20$	8

Table 4.1: The QoE metrics weight we considered [3]. Each metric is a variant of Equation 4.5.

It's important to note that [2] introduces additional QoE metrics, such as QoE_{log} and QoE_{hd}. The QoE_{log} metric reflects the idea that, for some users, the perceived quality improvement lessens at higher bitrates—a concept also utilized in the work by [52]. Conversely, QoE_{hd} emphasizes High Definition (HD) quality by assigning lower scores to non-HD streams and higher scores to HD streams.

These metrics differs by varying $c(u_k)$ and μ as depicted in Table 4.1.

4.3 Video Codecs in Video Streaming

Video codecs are essential components in video streaming, as they compress and encode video data for efficient transmission over networks, then decode it for playback on the user's device. Without codecs, transmitting raw video data would require substantial bandwidth, resulting in excessive data consumption and latency. In streaming applications, codecs strike a balance between video quality, file size, and computational efficiency, making them critical for delivering a seamless viewing experience, especially in adaptive streaming and low-latency environments.

4.3.1 Overview of Video Codec Functionality

A video codec operates by analyzing video frames to identify and compress redundant data, which reduces file size without significantly impacting visual quality. Codecs use two main types of compression: *intra-frame* and *inter-frame* compression. Intra-frame compression reduces redundancy within a single frame, whereas inter-frame compression identifies similarities between successive frames to reduce redundancies across a sequence. By employing both compression types, video codecs maintain video quality while minimizing file size and bandwidth requirements.

The codec processes video data in real-time to balance quality and bandwidth based on current network conditions. For adaptive bitrate streaming protocols like DASH and HLS, multiple encoded versions of each video segment at different bitrates are generated and stored on the server. The client then dynamically selects the version that matches its available bandwidth, ensuring optimal quality

while avoiding buffering interruptions.

4.3.2 Common Video Codecs in Streaming

Several video codecs are widely used in video streaming due to their balance of efficiency, compatibility, and quality:

- **H.264/AVC (Advanced Video Coding):** is one of the most popular codecs in video streaming, known for its high compression efficiency and wide compatibility across devices and platforms. It achieves a balance between quality and file size, making it suitable for various streaming applications, from live broadcasting to on-demand content. H.264 is widely supported in both hardware and software, providing versatility across different environments [53].

- **H.265/HEVC (High Efficiency Video Coding):** also known as HEVC, is the successor to H.264, offering up to 50% better compression efficiency while maintaining similar quality. This codec is ideal for high-definition (HD) and ultra-high-definition (UHD) content, as it can reduce file sizes, thus saving bandwidth. However, H.265's higher computational requirements may limit its use in devices with low processing power or energy constraints, such as in mobile and battery-operated systems [54].

- **VP9:** Developed by Google, is an open-source codec that provides similar compression efficiency to H.265 without licensing fees, making it popular for web-based streaming platforms, such as YouTube. VP9 is also optimized for high-definition content and is increasingly supported across modern devices and browsers, although it may lack the same hardware support as H.264 and H.265 [55].

- **AV1:** is a royalty-free codec developed by the Alliance for Open Media, designed to offer higher compression efficiency than both H.265 and VP9. AV1 is optimized for internet-based streaming and is well-suited for environments where bandwidth is constrained. While AV1 is still gaining support in hardware, its adoption is expected to increase as streaming services and device manufacturers incorporate this efficient, cost-effective codec [56].

Codec	Compression Efficiency	Computational Demand
H.264	Moderate	Moderate
H.265	High	High
VP9	High	Moderate-High
AV1	Very High	High

Table 4.2: Comparison of Common Video Codecs in Streaming.

Selecting an appropriate codec is crucial in adaptive streaming applications, as it directly impacts the user experience, bandwidth usage, and computational load. In adaptive bitrate streaming, such as DASH or HLS, the choice of codec affects both the storage requirements and streaming efficiency. High-compression codecs like H.265 and AV1 can reduce bandwidth usage and enhance video quality, particularly for high-definition and ultra-high-definition content, but they may require more processing power. Codecs such as H.264, with lower computational demands, are often favored for live streaming, low-latency applications, or devices with limited processing capacity.

The choice of codec can also influence Quality of Experience (QoE) by determining video quality at different network speeds and ensuring compatibility across user devices. For instance, H.264's broad compatibility makes it a safe choice for diverse streaming applications, whereas H.265 and AV1 may be preferred in high-quality streaming scenarios where higher compression can reduce data consumption without compromising visual fidelity.

For mobile robotics, which may operate in environments with limited or unstable connectivity, efficient codecs can play a critical role in transmitting video with minimal latency and maintaining quality under bandwidth constraints. The ability to adaptively stream video at different quality levels in real-time enhances the robot's ability to interpret visual data, supporting applications such as navigation, remote operation, and environmental monitoring.

4.4 Video Streaming Standards

In the context of video streaming for mobile robotics, several protocols enable efficient and reliable video transmission over networks with varying bandwidth and latency constraints. Among the most widely used are the Dynamic Adaptive Streaming over HTTP (DASH) protocol, HTTP Live Streaming (HLS) protocol, and Real-Time Communication (RTC) protocol. Each protocol has unique characteristics that address different aspects of transmission, such as adaptive bitrate, latency, and connection stability.

4.4.1 Dynamic Adaptive Streaming over HTTP (DASH)

Dynamic Adaptive Streaming over HTTP (DASH) [57] is the standard adopted for video streaming. YouTube, Netflix, to name just the most famous streamers, adopt such a standard. DASH is also used for 360 degree video streaming and for volumetric or 6DoF video streaming. As of today, more than half of the global Internet traffic is due to video contents [58]. DASH is designed to optimize video delivery in varying network conditions, allowing a seamless viewing experience by adjusting video quality in real-time.

The DASH standard segments video content into small chunks, each encoded

at multiple bitrates and resolutions. These segments are served over standard HTTP servers, and a client device can dynamically select the most appropriate segment quality based on current bandwidth, device capabilities, and network stability. A key component in DASH is the *Media Presentation Description (MPD) file*, called the *manifest* that provides metadata about the stream, including segment URLs, available bitrates, and timing information.

One of DASH's strengths is that it gives clients control over bitrate selection, making it highly flexible and compatible with a wide range of network and device conditions. This design allows the protocol to deliver high-quality video content while minimizing buffering and interruptions, making it suitable for applications like Video on Demand (VoD) and live streaming. DASH has become a fundamental protocol for adaptive streaming, supported across various devices and platforms, and continues to evolve to meet the demands of modern video delivery, including integration with emerging transport protocols like HTTP/3 over QUIC for enhanced performance on unstable networks.

4.4.2 HTTP Live Streaming (HLS)

HTTP Live Streaming (HLS) [59] is an adaptive bitrate streaming protocol developed by Apple to deliver high-quality video content over the internet, particularly optimized for devices in the Apple ecosystem. As DASH, it allows for dynamic adjustment of video quality based on network conditions, enabling smooth playback with minimal buffering and interruptions, even in fluctuating bandwidth environments.

HLS works by dividing video content into small segments, typically between 2 to 10 seconds in duration, each encoded at multiple bitrates and resolutions. A client device retrieves these segments sequentially based on current network per-

formance, choosing the bitrate that best balances quality and buffering avoidance. The protocol relies on an M3U8 playlist file, a manifest that contains URLs and metadata for each video segment, guiding the client's playback decisions.

As DASH, HLS has evolved to include support for protocols like HTTP/3 over QUIC, which enhances streaming performance in networks prone to latency and packet loss. While HLS is widely supported on iOS devices, its compatibility has since extended to other platforms.

4.4.3 WebRTC

Web Real-Time Communication (WebRTC) [60] differs fundamentally from streaming protocols like DASH and HLS in that it enables direct, peer-to-peer communication between devices, eliminating the need for a central server to manage media delivery. Whereas DASH and HLS depend on HTTP servers to store and distribute video segments to clients, WebRTC establishes a direct link between users' devices, allowing for low-latency, bidirectional streaming of audio, video, and data in real time.

The serverless design of WebRTC makes it especially suitable for applications where instant, interactive communication is critical, such as video conferencing, teleoperation, and live collaboration tools. This direct connection minimizes latency by avoiding the buffering and adaptive bitrate adjustments typical of DASH and HLS, which are designed to optimize video delivery over varying network conditions and often introduce delay.

In WebRTC, peer-to-peer connections are established through a signaling process, which uses STUN (Session Traversal Utilities for NAT) and, if necessary, TURN (Traversal Using Relays around NAT) servers only to handle initial connection setup, network traversal, and firewalls. Once a connection is established,

media flows directly between peers without relying on an intermediate server for content delivery. When more than two clients join a WebRTC call, the peer-to-peer setup can start to struggle. As more clients join, each device has to send and receive streams for every other participant, which can quickly overload both bandwidth and processing power. To solve this problem, many WebRTC systems use something called a Selective Forwarding Unit (SFU). An SFU acts as a smart relay for the media streams, so instead of each device connecting directly to every other device, they all connect to the SFU. The SFU then selectively forwards each stream to the right client, adjusting the quality and the number of streams based on needs and network capacity.

While DASH and HLS are optimized for streaming prerecorded and live content to multiple viewers with adaptive quality based on network conditions, WebRTC is tailored to scenarios requiring immediate, low-latency media exchange.

4.5 Adaptive Bitrate Algorithms

Adaptive bitrate (ABR) algorithms are essential in video streaming, enabling dynamic adjustment of video quality to ensure smooth playback despite fluctuations in network conditions. These algorithms optimize the user experience by balancing video quality and buffering to adapt to changing bandwidth, thereby minimizing playback interruptions and maximizing visual quality. In the literature, ABR algorithms are generally classified into two primary categories: rate-based (RB) and buffer-based (BB) algorithms [2]. Each approach offers unique strengths but also presents limitations, leading to interest in hybrid models that incorporate both throughput prediction and buffer occupancy.

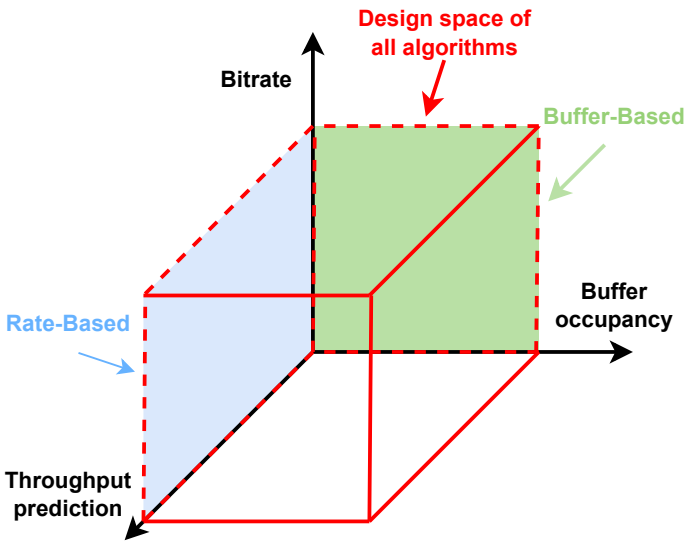

Figure 4.4: Design space of algorithms for the video adaptation problem [2]

4.5.1 Rate-Based Algorithms

Rate-based (RB) algorithms select video bitrate primarily based on real-time or predicted network throughput. By estimating the available bandwidth, RB algorithms choose the highest possible bitrate that the current network conditions can support without causing buffering. The fundamental idea behind RB algorithms is to continuously monitor network throughput and dynamically select a bitrate that aligns with available bandwidth. This approach is well-suited to network environments where bandwidth fluctuations are minimal or gradual, as throughput predictions can remain relatively stable.

However, they can struggle in networks with abrupt bandwidth changes or high variability. When bandwidth suddenly decreases, a rate-based algorithm may not respond quickly enough, leading to playback interruptions or buffering. Additionally, relying solely on throughput predictions makes RB strategies susceptible to

inaccuracies in bandwidth estimation, particularly when historical data is sparse or unreliable.

4.5.2 Buffer-Based Algorithms

Buffer-based (BB) algorithms, in contrast, make bitrate decisions based on the level of video data buffered at the client. This approach assumes that maintaining a stable buffer is critical to avoid playback interruptions, particularly in environments with highly variable bandwidth. The client continuously monitors buffer occupancy and adjusts the bitrate accordingly: when the buffer is nearly full, the algorithm may increase the bitrate to enhance video quality, while a low buffer level prompts a decrease in bitrate to prevent interruptions.

Buffer-based strategies are effective in handling abrupt network fluctuations since they react directly to buffer levels rather than throughput predictions. This design makes BB algorithms more resilient in networks with frequent or unpredictable bandwidth shifts. However, BB algorithms may select bitrates that exceed the available network bandwidth, especially when buffer levels are high, leading to congestion and potential rebuffering. Furthermore, because BB algorithms do not account for network throughput, they may choose overly conservative bitrates in situations where higher quality could be sustained, thus limiting video quality unnecessarily.

4.5.3 Hybrid Algorithms: Combining Rate-Based and Buffer-Based Approaches

While both RB and BB algorithms provide viable strategies for bitrate adaptation, each approach has inherent limitations when used in isolation. An ideal

adaptive bitrate algorithm would leverage both buffer occupancy and throughput prediction, thereby drawing from a broader design space (see Fig. 4.4). By combining the strengths of both approaches, hybrid algorithms can optimize video quality and stability more effectively than single-parameter strategies.

In a hybrid model, the algorithm assesses both the buffer level and the estimated throughput before making a bitrate selection. For example, if both the buffer level and throughput are high, the algorithm can safely choose a higher bitrate to enhance video quality. Conversely, when the buffer is low and throughput predictions indicate limited bandwidth, the algorithm reduces the bitrate to maintain smooth playback. This integrated approach enhances the responsiveness of hybrid algorithms to dynamic network conditions and improves their adaptability in balancing video quality with rebuffering control.

Chapter 5

Real-Time MPC Controller

This chapter presents the design and implementation of a new Real-time Adaptive Bitrate (ABR) algorithm, which is based on Model Predictive Control (MPC) and Dynamic Programming (DP) [61].

5.1 Related work

In [62] a Buffer-Based (BB) approach is presented to select the most appropriate bitrate during video playback. Such a strategy is designed to ensure smooth video playback by aiming at keeping the buffer occupancy above a predefined threshold. This algorithm is particularly effective in maintaining a high QoE for users, even during network fluctuations, due to the fact that it prioritises the buffer occupancy and adjusts the bitrate accordingly. Another BB technique is BOLA [52], which leverages Lyapunov optimization to select the best bitrate based only on the buffer occupancy.

Beside buffer occupancy, RobustMPC [2] also takes into account throughput predictions to determine the best bitrate that maximises a given QoE metric based on previous errors between predicted and observed throughputs.

The first work leveraging Deep Reinforcement Learning to design an ABR algorithms is Pensieve [3]. In particular, a neural network model is trained to learn the appropriate bitrates for future video chunks based on the data collected from individual client video players.

Another learning-based approach is proposed in [63], which employs Bayesian Neural Networks (BNN) combined with an MPC-based ABR algorithm to predict the probability distribution of the future throughput based on throughput of previously downloaded chunks to better adapt to dynamic networks and users.

To combine the advantages of buffer-based and learning-based ABR approaches, the authors of [64] introduce *Stick*. This ABR algorithm employs a Deep Reinforcement Learning (DRL) method to compute the buffer-bound needed by the buffer-based approach to maximise the QoE. This way, computational improvements are obtained wrt previous state-of-the-art techniques. In [65], a Meta Reinforcement Learning (Meta-RL) based ABR is designed. The algorithm rapidly adapts its control policy to changing network throughput dynamics by separating the inference of throughput dynamics from the universal control mechanism. Such an approach utilises a model-free system framework, consisting of a probabilistic latent encoder and a policy network conditioned on latent variables, to meta-learn the ABR policy.

All the aforementioned techniques are rather impractical when it comes to an actual implementation in a real-time streaming scenario. This is mainly due to the computational overhead introduced, which is usually neglected in simulated environments. The proposed RT-MPC aims at bridging this gap since it is particularly suitable for realistic implementations due to the low processing time required.

5.2 The proposed approach

At this point, the proposed control strategy to control an Adaptive Bitrate system can be presented.

5.2.1 Considerations on improving the QoE

As already explained in Section 4.2.1, frequent changes of bitrate levels adversely impact the QoE (*Smoothness penalty term* in (4.5)). When a change in the video level would contribute to a considerable enhancement in the visual quality and a significant portion of the video has elapsed without level changes, it might be advantageous to switch the video level to improve the QoE by paying a small penalty in terms of smoothness. This is based on the insight that isolated video-level switches might end up being beneficial for a user's QoE even at the cost of reduced smoothness.

To exploit this observation, we introduce the *Smoothness Window* which is a vector of size w defined as follows:

$$\mathbf{W} = [u_{s-w+1}, u_{s-w+1}, u_{s-w+2}, ...u_s]. \tag{5.1}$$

containing the last values of coding levels u belonging to the window. At the beginning, when filling the window vector $\mathbf{W}$, if the index $s - w + i < 0$ then $u_{s-w+i} = u_0$.

The idea is to modify the QoE (4.5) by adding a new component, named the *Smoothness window penalty*, that takes into account how the video levels have changed in the past window. This new added component penalizes frequent video level transitions over time, which is known to be detrimental to the user's

experience [66].

Thus, the QoE to be maximised by the RT-MPC controller over the prediction window H_P is defined as follows:

$$\begin{aligned} QoE_s^{s+H_P} = & \underbrace{\sum_{k=s}^{s+H_P} c(u_k)}_{\text{Quality reward}} - \lambda_1 \underbrace{\sum_{k=s}^{s+H_P-1} |c(u_{k+1}) - c(u_k)|}_{\text{Smoothness penalty}} \\ & - \mu_1 \underbrace{\sum_{k=s}^{s+H_P} r_s}_{\text{Rebuffering penalty}} - \beta \underbrace{\sum_{k=0}^{W-1} |u_{s-k} - u_{s-k-1}|}_{\text{Smoothness window penalty}} \end{aligned} \tag{5.2}$$

where λ_1, μ_1, and β are non-negative weighting parameters.

5.2.2 Considerations on bandwidth estimation

The literature presents several approaches to estimate the network bandwidth [67].

The analysis carried out in [2] compares different approaches for bandwidth estimation. Our approach starts by assuming that the network bandwidth is an unpredictable stochastic process. While some bandwidth estimators demonstrate accuracy when the network is in stationary conditions, the estimation becomes less accurate when either the bandwidth varies abruptly or the forecast horizon becomes longer. We assume the following bandwidth estimator:

$$\hat{B}_s^{s+1} = \gamma B_s \tag{5.3}$$

In other words, the bandwidth at the next step is estimated as a fraction $0 < \gamma \leq 1$ of the current measured bandwidth, which is a sort of worst-case assumption when performing optimization.

5.2.3 The controller

Ideally, an ABR algorithm should have a control strategy that starts to download the next segment instantaneously when the previous download is ended. In the reality, the controller has a computational delay w_s that retards the download of the next segment. The majority of approaches presented in 5.1 neglects this fact, focusing only on the QoE maximization problem. Yin et al. in [2] raise concerns about computational overhead developing the idea of an offline approach called *FastMPC* where they enumerated possible scenarios and create a table indexing the optimal decision for each scenario. In theory this would be a good method but in practical it has some issues that although they have been addressed, have not been resolved:

- Inefficient in case of an high dimensional state space;
- Memory overhead needed to store the table of all possible cases;
- Offline computation cost could be higher when is needed to rerun the optimization in case of changing the operating conditions.

Starting from scratch by analyzing the problem, we observed that the optimal bitrate selection involves traversing a tree structure as the one illustrated in Figure 5.1 and identifying the path that maximizes the solution. An exhaustive exploration of the tree along the prediction horizon would require $\mathcal{O}(|\mathcal{L}|^{H_P})$ states while evaluating each state would involve an $\mathcal{O}(H_P)$ search plus some constant time operations. Thus, the overall computational cost of the algorithm would be $\mathcal{O}(|\mathcal{L}|^{H_P} H_P)$, which is exponential in the prediction horizon.

In light of the fact that the proposed optimization problem is a multi-stage decision process, we exploit the Bellman's optimality principle [68] to reformulate

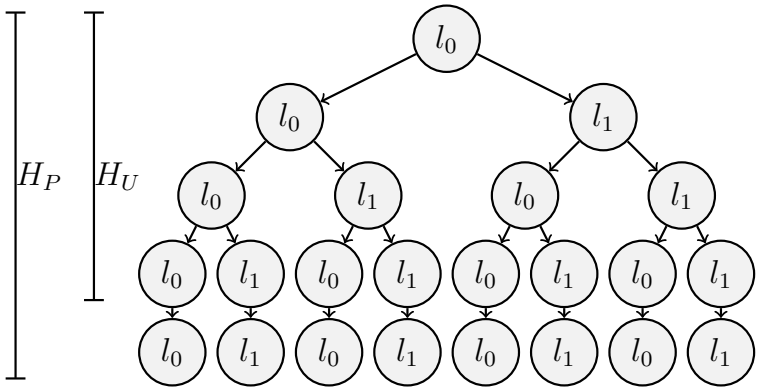

Figure 5.1: The recursive tree for computing the optimal sequence of bitrate considering $\mathcal{L} = \{l_0, l_1\}$, $H_P = 4$, $H_U = 3$.

the optimization problem as follows:

$$J^* = \max_{u_s} \left[QoE_s^{s+1} + \max_{u_{s+Hp-1}} \left[QoE_{s+Hp-1}^{s+Hp} \right] \right] \tag{5.4}$$

By using Dynamic Programming to solve (5.4), it is possible to turn the exponential cost to a polynomial one, specifically achieving a complexity of $\mathcal{O}(|\mathcal{L}|{H_P}^2)$. By examining the reward function (5.2), it is clear that the *Quality Reward*, *Smoothness Penalty* and *Smoothness Window Penalty* terms depend only on the input sequence from s to $s + H_p$. Given their independence from the system's state, these terms can be computed a priori. In light of this key observation, during the initialization phase, the RT-MPC algorithm generates a table of potential rewards for all possible paths in the tree (Figure 5.1), thus further reducing the computational cost to $\mathcal{O}(|\mathcal{L}|H_P)$.

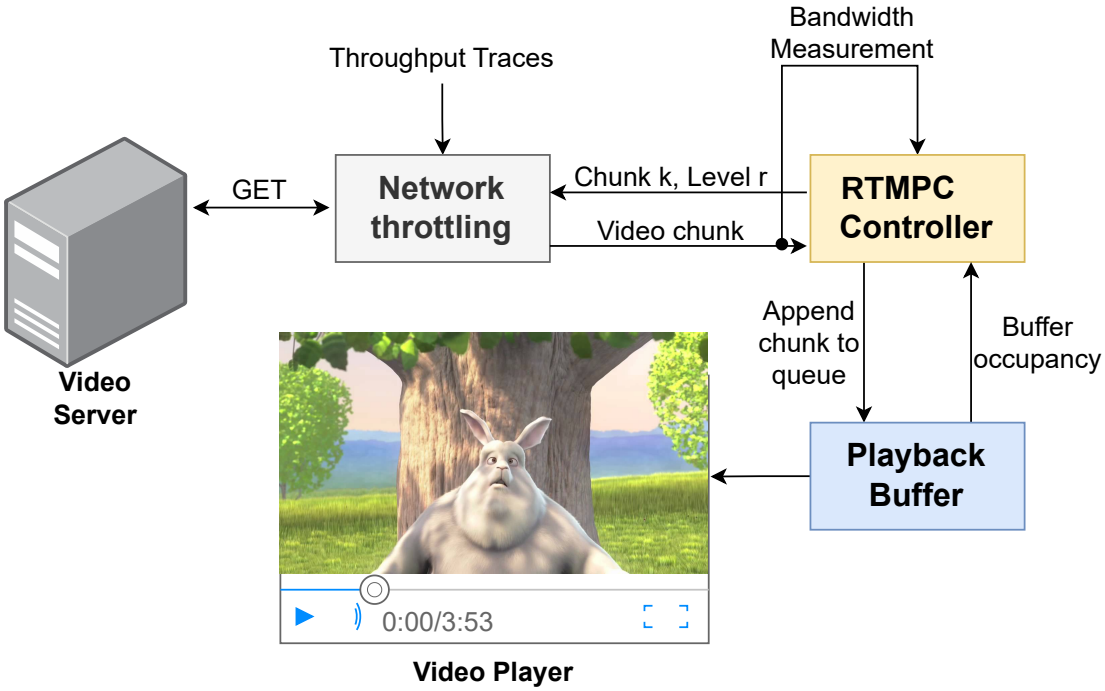

Figure 5.2: Testbed setup representation

5.3 Experimental results

This section will provide an overview of the experimental results.

5.3.1 Testbed setup

Let $\mathcal{L} = \{300, 750, 1200, 1850, 2850, 4300\}$ kbps be the set of available bitrate levels. The duration of each video chunk is set to $\tau = 4$ seconds, and the total number of video chunks is 48, which corresponds to a video duration of 192 seconds. The calibration of the controller parameters has been carried out utilising Optuna [69], which is tool for parameter tuning based on the maximization of a reward function. The resulting values for the parameters are as follows: $H_p = 4$, $H_u = 4$, $W = 4$, $c(l_s) = l_s/1000$, $\gamma = 1.45$, $\beta = 7.21$, $\Delta_u = 3$, $\lambda_1 = 2.48$ and $\mu_1 = 0.33$.

Regarding the QoE_{lin} (4.5) used for performance comparison, we set $\lambda = 4.3$ and $\mu = 1$ as in [2]. Finally, the buffer capacity is limited to 1 minute. All experiments were performed on a machine running Ubuntu 22.04 with 32GB of RAM and an Intel i7 8th generation CPU.

5.3.2 Scenarios

The goal is to investigate different bitrate adaptation techniques under realistic network conditions. To the purpose, the following pre-existing datasets of network bandwidth available over the Internet have been used:

- **Federal Communications Commission (FCC)** [70]: the dataset consists of more than 1 million sets of throughput measurements during a 5s interval. We extract throughput traces of the same server and client IP address and concatenate these to match the length of the video. For experiments, we randomly pick 1000 concatenated traces.

- **High Speed Downlink Packet Access (HSDPA)** [71]: the dataset consists of 30 minutes of continuous 1s measurements of video streaming throughput of a moving device in Telenor's 3G/HSDPA mobile wireless network in Norway. We randomly pick 1000 throughput traces.

- **Oboe dataset** [72]: it is a set of bandwidth traces of real video streaming sessions. The data corresponds to 500 video sessions. For each session, the bandwidth observed for each downloaded video chunk is provided.

5.3.3 Adaptation algorithms

We compare RT-MPC against the following algorithms, which represent the state-of-the-art in bitrate adaptation techniques: Buffer-Based (BB) [62], Rate-Based

(RB), BOLA [52], RobustMPC [2], Pensieve [3], BayesMPC [63], and Merina [65]. Furthermore, to provide an absolute reference for comparison, we also include the results obtained using the offline optimal strategy, which gives the best achievable solution. The offline optimal strategy represents the maximum QoE that an oracle policy with complete and perfect knowledge of future network throughput could achieve. This strategy is obtained through dynamic programming with complete information about future throughput, which is never available in practice. It is therefore to be considered as a reference in the testing phase.

5.3.4 Results

Figure 5.3 illustrates the Cumulative Distribution Function (CDF) of the average QoE_{lin} values obtained for each video chunk using the FCC dataset. This CDF allows us to visually assess the performance distribution across different approaches by showing the proportion of chunks achieving or exceeding a given QoE_{lin} score. In examining the figure, we observe that, aside from the offline optimal strategy, represented by the red dashed line on the far right of the CDF curve, RT-MPC (shown as the orange solid line) achieves the highest average QoE_{lin} among the other adaptive bitrate algorithms considered. This suggests that RT-MPC consistently maximizes quality perception as measured by QoE_{lin}, positioning it close to the optimal solution while operating in real-time conditions.

Further validation of RT-MPC's performance in delivering high QoE can be observed in Figure 5.4. This figure breaks down the average contributions of the first three components of the QoE_{lin} metric, as defined in Equation (4.5), for each algorithm analyzed, once again utilizing the FCC dataset. Specifically, RT-MPC demonstrates the highest *Quality reward*, meaning that it maintains superior video quality levels across chunks compared to the other methods. This achievement is

balanced with a low *Rebuffering penalty*, indicating that the algorithm effectively minimizes interruptions in playback. However, this enhanced performance comes with a trade-off in the form of a slightly higher *Smoothness penalty* relative to the other algorithms. The elevated smoothness penalty is largely due to RT-MPC's use of a conservative worst-case bandwidth estimator, which occasionally results in bitrate fluctuations as it adjusts for potential variations in network capacity. Nonetheless, this trade-off is minor and does not detract significantly from the overall QoE. Additional tests performed using the HDSPA and Oboe datasets showed similar patterns, underscoring RT-MPC's robust performance across diverse network conditions, though we omit these results here for brevity.

Overall, RT-MPC achieves notable improvements in average QoE values, with gains ranging between 3% and 6% across different segments of the FCC dataset. These improvements demonstrate RT-MPC's effectiveness in enhancing user experience in adaptive streaming contexts, where maintaining high QoE is a critical objective.

A comparison of computational times across the different datasets is provided in Table 5.1. The results in this table reveal that RT-MPC delivers a substantial reduction in computation time compared to the other algorithms. This improvement highlights the algorithm's capability to make adaptive decisions efficiently, meeting the demands of real-time processing with minimal delay. Furthermore, RT-MPC's computational performance is characterized by a variance close to zero, as shown in Table 5.1, underscoring its reliability in achieving near-deterministic precision in execution times. Such consistency makes RT-MPC an ideal candidate for live video streaming applications, where the processing time per chunk is constrained to intervals as short as half a second. This deterministic performance is crucial for real-time streaming scenarios, where the timeliness of bitrate

adaptation directly impacts user experience.

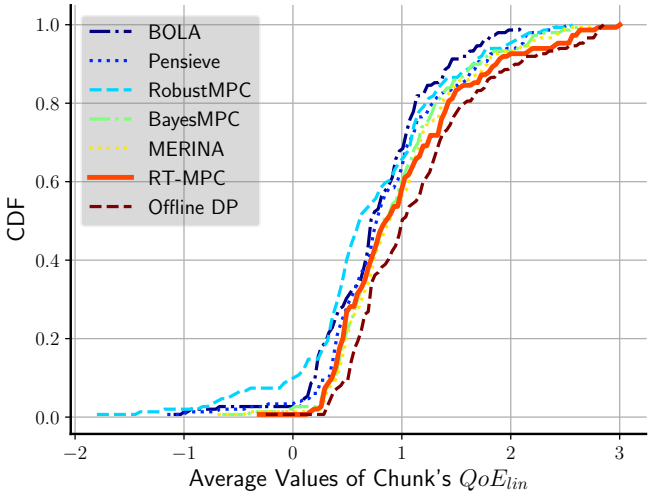

Figure 5.3: CFD of the average values of chunk's QoE_{lin} over the FCC dataset.

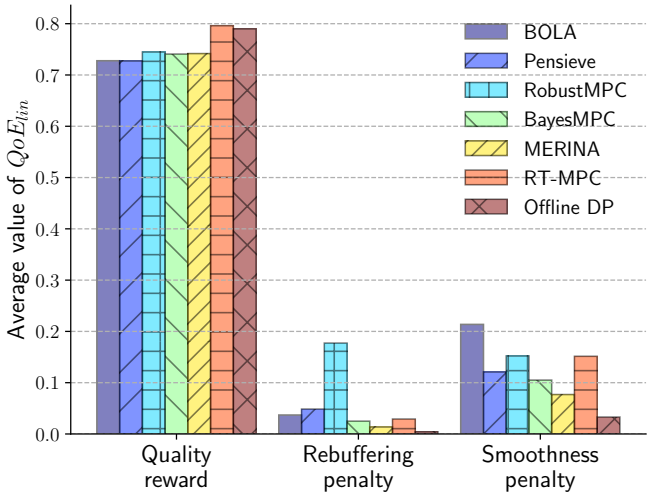

Figure 5.4: Comparison of the single terms in the QoE_{lin} metric over the FCC dataset.

5.4 Concluding remarks

In this chapter, we have presented RT-MPC, a novel Adaptive Bitrate (ABR) algorithm designed specifically to enhance video quality selection in streaming

	FCC	**HSDPA**	**Oboe**
Pensieve	$\mu = 1.79$ $\sigma^2 = 1.32$ $min = 0.79$ $max = 108.84$	$\mu = 2.84$ $\sigma^2 = 1.66$ $min = 0.82$ $max = 122.64$	$\mu = 4.89$ $\sigma^2 = 1.46$ $min = 0.90$ $max = 167.84$
RobustMPC	$\mu = 3.58$ $\sigma^2 = 5.53$ $min = 0.02$ $max = 35.40$	$\mu = 6.13$ $\sigma^2 = 7.04$ $min = 0.02$ $max = 40.57$	$\mu = 10.95$ $\sigma^2 = 8.62$ $min = 0.02$ $max = 38.37$
BayesMPC	$\mu = 39.80$ $\sigma^2 = 3.18$ $min = 37.71$ $max = 211.74$	$\mu = 42.31$ $\sigma^2 = 3.44$ $min = 36.12$ $max = 325.89$	$\mu = 56.67$ $\sigma^2 = 3.18$ $min = 37.71$ $max = 190.33$
Merina	$\mu = 1.04$ $\sigma^2 = 1.84$ $min = 0.96$ $max = 157.68$	$\mu = 2.77$ $\sigma^2 = 2.31$ $min = 0.90$ $max = 156.80$	$\mu = 3.08$ $\sigma^2 = 2.25$ $min = 1.39$ $max = 227.68$
RT-MPC	$\mu = 0.10$ $\sigma^2 = 0.02$ $min = 0.05$ $max = 0.21$	$\mu = 0.38$ $\sigma^2 = 0.04$ $min = 0.11$ $max = 0.55$	$\mu = 0.28$ $\sigma^2 = 0.02$ $min = 0.07$ $max = 0.42$

Table 5.1: Computational time comparison over different datasets. Values are expressed in ms.

applications. By focusing on maximizing the user's Quality of Experience (QoE), RT-MPC addresses the challenges posed by fluctuating network conditions and the need for real-time decision-making in video streaming environments. To tackle the computational complexity typically associated with Model Predictive Control (MPC) in ABR applications, RT-MPC integrates Bellman Dynamic Programming, alongside constraints on maximum encoding level changes. These constraints effectively reduce the scope of optimization, making it computationally feasible to implement the algorithm in real time without sacrificing performance or quality.

The effectiveness of RT-MPC has been rigorously evaluated against several leading ABR algorithms widely recognized in the literature, demonstrating sub-

stantial improvements not only in QoE metrics but also in computational efficiency, both of which are crucial for real-time streaming applications. Specifically, RT-MPC significantly reduces the average, standard deviation, and maximum computation times by several orders of magnitude compared to traditional ABR approaches. This improvement is pivotal, as it enables the algorithm to make rapid adjustments to video quality, thereby mitigating buffering events and maintaining smooth playback, which are central to enhancing the user experience in dynamic network environments.

Moreover, the design of RT-MPC incorporates real-time adaptability, ensuring that the algorithm can effectively respond to variations in network bandwidth and user device constraints. The integration of predictive control and optimization-based adjustments allows RT-MPC to balance video quality and stability, resulting in fewer quality fluctuations and a more consistent viewing experience. This balance is particularly relevant for applications where low latency and seamless video playback are essential, such as live streaming, remote collaboration, and virtual reality environments.

In conclusion, RT-MPC sets a new benchmark in ABR algorithm design by combining the theoretical rigor of Model Predictive Control with practical constraints set by real-time performance. Its ability to deliver superior QoE while significantly reducing computation time makes RT-MPC a promising solution for future streaming applications that demand both high responsiveness and high-quality media delivery. The findings presented in this chapter underscore the potential of RT-MPC to advance the state of ABR technology, paving the way for further research into optimization techniques that enhance real-time adaptability and computational efficiency in video streaming.

Chapter 6

Conclusions and Future Research Directions

The thesis has explored the field of autonomous mobile robots. In the first part, we focused on mobile robotics, specifically developing a safe and fault-tolerant controller for industrial differential drive vehicles. The second part shifted to the decentralized control of drone swarms, while also addressing video streaming systems and presenting a novel Adaptive Bitrate (ABR) algorithm, RT-MPC, which outperforms existing solutions.

The contributions of this thesis are:

- **Development of a Fault-Tolerant Control System for Industrial Vehicles**: A novel system designed to detect and mitigate rotary speed sensor failures in differential drive industrial vehicles, ensuring safe operation even in the event of sensor faults, and fully compliant with European safety standards.
- **Formalization and Development of a Framework for Decentralized Control of Bandwidth-Aware Drone Swarms**: A framework that inte-

grates decentralized control strategies to optimize video streaming quality in drone swarms, considering both drone altitude and available bandwidth.

- **Development of a New MPC-Based Adaptive Bitrate Algorithm**: A new Adaptive Bitrate (ABR) algorithm based on Model Predictive Control (MPC), which outperforms current state-of-the-art solutions in terms of Quality of Experience (QoE) and computational efficiency. Our tailored solution reduces the problem's computational complexity from exponential to polynomial time.

Future Research

While this thesis has made significant strides in addressing key issues in both autonomous robotics and video streaming, several avenues for future research remain. In the area of fault-tolerant control systems for industrial vehicles, further studies could explore the integration of machine learning techniques to improve sensor fault detection and prediction, especially in environments with more complex fault patterns. Additionally, extending this work to multi-vehicle fleets could provide valuable insights into optimizing fault detection and vehicle coordination at scale.

For drone swarm systems, future research could investigate more advanced algorithms that combine NMPC with machine learning-based path planning and obstacle avoidance. These approaches could further enhance the autonomy of UAVs in highly dynamic and unpredictable environments. Furthermore, the integration of real-time video analytics with drone systems could be explored, allowing for intelligent decision-making based on video streams, such as object detection or event recognition, to improve the effectiveness of drone-based missions.

In video streaming, the RT-MPC algorithm could be expanded to handle more complex network conditions. Additionally, incorporating Quality of Service (QoS) metrics into the algorithm's optimization framework could lead to further improvements in overall streaming performance. Exploring the application of RT-MPC in scenarios with low-latency requirements, such as remote surgeries or autonomous vehicle communication, could open new possibilities for real-time, high-quality media delivery.

Ultimately, the intersection of autonomous robotics and real-time communication is a rapidly evolving field with vast potential for innovation. The methodologies and solutions presented in this thesis lay the groundwork for further exploration and development, with the goal of advancing the capabilities of autonomous systems and enhancing the user experience in complex, dynamic environments. Future research in these areas will undoubtedly contribute to the continued evolution of both autonomous technologies and communication systems, making them more reliable, efficient, and adaptable to a wide range of real-world applications.

Bibliography

[1] O. Hohlfeld, E. Pujol, F. Ciucu, A. Feldmann, and P. Barford, "A qoe perspective on sizing network buffers," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14, (New York, NY, USA), p. 333–346, Association for Computing Machinery, 2014.

[2] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pp. 325–338, 2015.

[3] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the conference of the ACM special interest group on data communication*, pp. 197–210, 2017.

[4] "Warehouse automation market by technology (agv/amr, as/rs, conveyors, sortation, order picking, automatic identification and data capture, palletizing & depalletizing, overhead systems, mro services and wms/wes/wcs), by industry (e-commerce, general merchandise, grocery, apparel, food & beverage, pharma, 3pl), by functions (inbound, picking, outbound), by geography – global forecast to 2025," 2020. Global market forecast and analysis for warehouse automation technologies, industries, and geographic regions.

[5] R. R. Murphy, *Disaster Robotics*. Cambridge, MA: MIT Press, 2014.

[6] M. A. Rezaei, G. Manfredi, V. A. Racanelli, L. De Cicco, and S. Mascolo, "Decentralized control of uav swarms for bandwidth-aware video surveillance using nmpc," in *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 947–954, 2024.

[7] James B. Rawlings and David Q. Mayne, *Model Predictive Control: Theory, Computation, and Design.* Madison, WI: Nob Hill Publishing, 2nd ed., 2009.

[8] V. A. Racanelli and S. Mascolo, "Safe and fault tolerant control of industrial differential drive vehicles," *IFAC-PapersOnLine*, vol. 58, no. 4, pp. 150–155, 2024.

[9] OSHA, "Occupational safety and health administration - fatality inspection data," 2023. `https://www.osha.gov/fatalities`.

[10] "Directive 2006/42/ec of the european parliament and of the council of 17 may 2006 on machinery, and amending directive 95/16/ec (recast) (text with eea relevance)," Jun 2006.

[11] "Iec 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems," 2010. Sets out a general approach to functional safety for electronic systems.

[12] K. Astrom, "Pid controllers-theory," *Design and Tuning*, 1995.

[13] I. Koren and C. M. Krishna, *Fault-tolerant systems.* Morgan Kaufmann, 2020.

[14] R. Isermann, *Fault-diagnosis systems: an introduction from fault detection to fault tolerance.* Springer Science & Business Media, 2005.

[15] M. Bourogaoui, H. B. A. Sethom, and I. S. Belkhodja, "Speed/position sensor fault tolerant control in adjustable speed drives–a review," *ISA transactions*, vol. 64, pp. 269–284, 2016.

[16] B. Tabbache, M. Benbouzid, A. Kheloui, and J.-M. Bourgeot, "Dsp-based sensor fault detection and post fault-tolerant control of an induction motor-based electric vehicle," *International Journal of Vehicular Technology*, vol. 2012, 2012.

[17] S. Fan and J. Zou, "Sensor fault detection and fault tolerant control of induction motor drivers for electric vehicles," in *Proceedings of the 7th International Power Electronics and Motion Control Conference*, vol. 2, pp. 1306–1309, IEEE, 2012.

[18] F. Zidani, D. Diallo, M. Benbouzid, and E. Berthelot, "Diagnosis of speed sensor failure in induction motor drive," in *2007 IEEE international electric machines & drives conference*, vol. 2, pp. 1680–1684, IEEE, 2007.

[19] H. Tang, Y. Chen, and A. Zhou, "Actuator fault-tolerant control for four-wheel-drive-by-wire electric vehicle," *IEEE transactions on transportation electrification*, vol. 8, no. 2, pp. 2361–2373, 2021.

[20] "En 62061: Safety of machinery – functional safety of safety-related electrical, electronic and programmable electronic control systems," 2005. European standard based on IEC 61508, specific to machinery safety.

[21] "Iso 13849-1: Safety of machinery – safety-related parts of control systems – part 1: General principles for design," 2015. Standard used to assess the safety performance level (PL) of control systems in machinery.

[22] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[23] M. Nordin and P.-O. Gutman, "Controlling mechanical systems with backlash—a survey," *Automatica*, vol. 38, no. 10, pp. 1633–1649, 2002.

[24] T. M. Cabreira, C. D. Franco, P. R. Ferreira, and G. C. Buttazzo, "Energy-Aware Spiral Coverage Path Planning for UAV Photogrammetric Applications," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3662–3668, 2018.

[25] S. Asadzadeh, W. J. de Oliveira, and C. R. de Souza Filho, "Uav-based remote sensing for the petroleum industry and environmental monitoring: State-of-the-art and perspectives," *Journal of Petroleum Science and Engineering*, vol. 208, p. 109633, 2022.

[26] B. Mishra, D. Garg, P. Narang, and V. Mishra, "Drone-surveillance for search and rescue in natural disaster," *Computer Communications*, vol. 156, pp. 1–10, 2020.

[27] X. Meng, W. Wang, and B. Leong, "Skystitch: A cooperative multi-uav-based real-time video surveillance system with stitching," in *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 261–270, 2015.

[28] S. P. H. Boroujeni and A. Razi, "Ic-gan: An improved conditional generative adversarial network for rgb-to-ir image translation with applications to forest fire monitoring," *Expert Systems with Applications*, p. 121962, 2023.

[29] J. Tang, G. Liu, and Q. Pan, "A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 10, pp. 1627–1643, 2021.

[30] F. Mehdifar, C. P. Bechlioulis, F. Hashemzadeh, and M. Baradarannia, "Prescribed performance distance-based formation control of multi-agent systems," *Automatica*, vol. 119, p. 109086, 2020.

[31] M. A. Rezaei, P. Bagheri, and F. Hashemzadeh, "Consensus tracking of multi-agent systems in the presence of byzantine agents using model predictive control," in *2022 8th International Conference on Control, Instrumentation and Automation (ICCIA)*, pp. 1–5, 2022.

[32] M. A. Rezaei, P. Bagheri, and F. Hashemzadeh, "Predictive consensus tracking of multi-agent systems in the presence of byzantine agents and connection loss of reference signals," *Optimal Control Applications and Methods*, vol. 45, no. 2, pp. 842–854, 2024.

[33] I. Medeiros, A. Boukerche, and E. Cerqueira, "Swarm-based and energy-aware unmanned aerial vehicle system for video delivery of mobile objects," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 1, pp. 766–779, 2021.

[34] H. Jiang, Y. Chang, L. Yang, X. Liu, and Y. He, "Cooperative Exploration of Heterogeneous UAVs in Mountainous Environments by Constructing Steady Communication," *IEEE Robotics and Automation Letters*, 2023.

[35] H. S. Munawar, A. W. Hammad, and S. T. Waller, "Disaster Region Coverage Using Drones: Maximum Area Coverage and Minimum Resource Utilisation," *Drones*, vol. 6, no. 4, p. 96, 2022.

[36] A. Bist and C. Singhal, "Efficient immersive surveillance of inaccessible regions using uav network," in *IEEE INFOCOM 2021-IEEE Conference*

on Computer Communications Workshops (INFOCOM WKSHPS), pp. 1–6, 2021.

[37] U. Choi and S. Lee, "Bandwidth-aware coverage path planning for swarm of uavs with aerial base station," in *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 360–365, 2023.

[38] M. Schwager, B. J. Julian, M. Angermann, and D. Rus, "Eyes in the sky: Decentralized control for the deployment of robotic camera networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1541–1561, 2011.

[39] A. Tahirovic and A. Astolfi, "A convergent solution to the multi-vehicle coverage problem," in *2013 American Control Conference*, pp. 4635–4641, IEEE, 2013.

[40] L. Wei, Z. Zhong, C. Lang, and Z. Yi, "A survey on image and video stitching," *Virtual Reality & Intelligent Hardware*, vol. 1, no. 1, pp. 55–83, 2019.

[41] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy, "Bandwidth estimation: metrics, measurement techniques, and tools," *IEEE network*, vol. 17, no. 6, pp. 27–35, 2003.

[42] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Congestion control for web real-time communication," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2629–2642, 2017.

[43] J. C. Leachtenauer and R. G. Driggers, *Surveillance and reconnaissance imaging systems: modeling and performance prediction*. Artech House, 2001.

[44] H. Huang, A. V. Savkin, and C. Huang, "Decentralized autonomous navigation of a uav network for road traffic monitoring," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 4, pp. 2558–2564, 2021.

[45] H.-S. Ahn, *Formation control*. Springer, 2020.

[46] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, "Elastic: A client-side controller for dynamic adaptive streaming over http (dash)," in *2013 20th International Packet Video Workshop*, pp. 1–8, IEEE, 2013.

[47] G. Manfredi, L. De Cicco, and S. Mascolo, "Synchronizing live video streaming players via consensus," in *2021 European Control Conference (ECC)*, pp. 1062–1067, IEEE, 2021.

[48] P. Le Callet, S. Möller, A. Perkis, K. Brunnström, S. Beker, K. De Moor, A. Dooms, S. Egger, M.-N. Garcia, T. Hoßfeld, *et al.*, *Qualinet white paper on definitions of quality of experience*. PhD thesis, Qualinet (www. qualinet. eu), 2013.

[49] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," *ACM SIGCOMM computer communication review*, vol. 41, no. 4, pp. 362–373, 2011.

[50] R. K. Mok, E. W. Chan, and R. K. Chang, "Measuring the quality of experience of http video streaming," in *12th IFIP/IEEE international symposium on integrated network management (IM 2011) and workshops*, pp. 485–492, IEEE, 2011.

[51] J. B. Husić and S. Baraković, "Multidimensional modelling of quality of experience for video streaming," *Computers in human behavior*, vol. 129, p. 107155, 2022.

[52] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," *IEEE/ACM transactions on networking*, vol. 28, no. 4, pp. 1698–1711, 2020.

[53] ISO/IEC and ITU-T, "Advanced Video Coding for Generic Audiovisual Services," 2014. Also known as MPEG-4 Part 10 or AVC (Advanced Video Coding).

[54] ISO/IEC and ITU-T, "High Efficiency Video Coding (HEVC)," 2015. Also known as HEVC (High Efficiency Video Coding).

[55] Google Inc., "VP9 Video Codec Specification," 2013. Available through the WebM Project.

[56] Alliance for Open Media, "AV1 Bitstream Decoding Process Specification," 2018. Version 1.0.0.

[57] ISO/IEC, "Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats," 2022. Standard.

[58] Sandvine, "2024 global internet phenomena report," *Report*, 2024.

[59] Pantos, Rod and Apple Inc., "HTTP Live Streaming." IETF Internet-Draft draft-pantos-hls-rfc8216bis-08, 2023. Work in progress.

[60] W3C and IETF, "WebRTC 1.0: Real-Time Communication Between Browsers." World Wide Web Consortium (W3C) Recommendation, 2021. W3C Recommendation.

[61] V. A. Racanelli, G. Manfredi, L. De Cicco, and S. Mascolo, "Real-time mpc for adaptive video streaming," *Proc. of IEEE Consumer Communications Networking Conference*, 2025. accepted.

[62] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the 2014 ACM conference on SIGCOMM*, pp. 187–198, 2014.

[63] N. Kan, C. Li, C. Yang, W. Dai, J. Zou, and H. Xiong, "Uncertainty-aware robust adaptive video streaming with bayesian neural network and model predictive control," in *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 17–24, 2021.

[64] T. Huang, C. Zhou, R.-X. Zhang, C. Wu, X. Yao, and L. Sun, "Stick: A harmonious fusion of buffer-based and learning-based approach for adaptive streaming," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 1967–1976, IEEE, 2020.

[65] N. Kan, Y. Jiang, C. Li, W. Dai, J. Zou, and H. Xiong, "Improving generalization for neural adaptive video streaming via meta reinforcement learning," in *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 3006–3016, 2022.

[66] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on quality of experience of http adaptive streaming," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.

[67] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in dynamic http streaming," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pp. 109–120, 2012.

[68] R. E. Bellman and S. E. Dreyfus, *Applied dynamic programming*, vol. 2050. Princeton university press, 2015.

[69] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[70] F. C. Commission, "Measuring broadband raw data releases." `https://www.fcc.gov/oet/mba/raw-data-releases`, 2023.

[71] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3g networks: analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*, pp. 114–118, 2013.

[72] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang, "Oboe: Auto-tuning video abr algorithms to network conditions," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pp. 44–58, 2018.

[73] "Directive 2006/42/ec of the european parliament and of the council of 17 may 2006 on machinery, and amending directive 95/16/ec (recast)," 2006. Official Journal of the European Union.

[74] "En 280: Mobile elevating work platforms - design calculations, stability criteria, construction - safety - examinations and tests," May 2013. Specifies requirements for mobile elevating work platforms (MEWPs).

[75] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback control of dynamic systems*, vol. 33. Pearson London, 2015.

[76] L. De Cicco, S. Mascolo, and V. Palmisano, "Feedback control for adaptive live video streaming," in *Proceedings of the second annual ACM conference on Multimedia systems*, pp. 145–156, 2011.

[77] G. Cofano, L. De Cicco, and S. Mascolo, "A hybrid model of adaptive video streaming control systems," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 6601–6606, IEEE, 2016.

[78] K. Mitra, A. Zaslavsky, and C. Åhlund, "Qoe modelling, measurement and prediction: A review," *arXiv preprint arXiv:1410.6952*, 2014.

[79] D. Tsolkas, E. Liotou, N. Passas, and L. Merakos, "A survey on parametric qoe estimation for popular services," *Journal of network and computer applications*, vol. 77, pp. 1–17, 2017.

[80] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proceedings of the 2016 ACM SIGCOMM Conference*, pp. 272–285, 2016.

[81] A. O. El Meligy, M. S. Hassan, and T. Landolsi, "A buffer-based rate adaptation approach for video streaming over http," in *2020 Wireless Telecommunications Symposium (WTS)*, pp. 1–5, IEEE, 2020.

[82] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan, "Mahimahi: accurate {Record-and-Replay} for {HTTP}," in *2015 USENIX Annual Technical Conference (USENIX ATC 15)*, pp. 417–429, 2015.

[83] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pp. 97–108, 2012.

[84] C. Qu, R. Singh, A. Esquivel-Morel, and P. Calyam, "Learning-based multi-drone network edge orchestration for video analytics," in *Proc. of IEEE INFOCOM 2022*, pp. 1219–1228, IEEE, 2022.

[85] M. Gasparini, J. C. Moreno-Escribano, and A. Monterroso-Checa, "Photogrammetric acquisitions in diverse archaeological contexts using drones: Background of the ager mellariensis project (north of córdoba-spain)," *Drones*, vol. 4, no. 3, p. 47, 2020.

[86] R. Shirey, S. Rao, and S. Sundaram, "Optimizing quality of experience for long-range uas video streaming," in *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, pp. 1–10, 2021.

[87] L. A. b. Burhanuddin, X. Liu, Y. Deng, U. Challita, and A. Zahemszky, "Qoe optimization for live video streaming in uav-to-uav communications via deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 5358–5370, 2022.

[88] M. Aloqaily, O. Bouachir, I. Al Ridhawi, and A. Tzes, "An adaptive uav positioning model for sustainable smart transportation," *Sustainable Cities and Society*, vol. 78, p. 103617, 2022.

[89] J. Sabzehali, V. K. Shah, Q. Fan, B. Choudhury, L. Liu, and J. H. Reed, "Optimizing Number, Placement, and Backhaul Connectivity of Multi-UAV Networks," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 21548–21560, 2022.

[90] T.-Y. Fan, F. Liu, J.-W. Fang, N. Venkatasubramanian, and C.-H. Hsu, "Enhancing situational awareness with adaptive firefighting drones: leveraging diverse media types and classifiers," in *Proceedings of the 13th ACM Multimedia Systems Conference*, pp. 279–286, 2022.

[91] C. Song, B. Han, X. Ji, Y. Li, and J. Su, "Ai-driven multipath transmission: Empowering uav-based live streaming," *IEEE Network*, 2023.

[92] N. González, M. Solera, F. Ruiz, C. Gijón, and M. Toril, "A quality of experience model for live video in first-person-view drone control in cellular networks," *Computer Networks*, p. 110089, 2023.