МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНЕ НЕКОМЕРЦІЙНЕ ПІДПРИЄМСТВО

«КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

Факультет аеронавігації, електроніки та телекомунікацій

Кафедра авіаційних комп'ютерно-інтегрованих комплексів

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач випускової кафедри

_____ Віктор СИНЄГЛАЗОВ

"____"_____2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСНИКА ОСВІТНЬОГО СТУПЕНЯ

«БАКАЛАВР»

Спеціальність 151 «Автоматизація та комп'ютерно-інтегровані технології»

Освітньо-професійна програма «Інформаційні технології та інженерія авіаційних комп'ютерних систем»

Тема: Інтелектуальне керування роєм БПЛА на основі моделі лідер–послідовник в умовах обмеженого зв'язку

Виконавець: здобувач вищої освіти Шемчук Павло Володимирович

Керівник: ст. викладач, к.т.н. Долгоруков Сергій Олегович

Нормоконтролер: _____Філяшкін М.К.

(підпис)

Київ - 2025

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

STATE NON-PROFIT ENTERPRISE

"KYIV AVIATION INSTITUTE"

Faculty of Air Navigation, Electronics and Telecommunications

Department of Aviation Computer-Integrated Complexes

ADMIT TO DEFENSE

Head of the Graduate Department

_____ Viktor SYNIEHLAZOV

"____"_____2025

QUALIFICATION WORK

(EXPLANATORY NOTE)

GRADUATE OF AN EDUCATIONAL DEGREE

"BACHELOR"

Specialty 151 "Automation and Computer-Integrated Technologies"

Educational and professional program "Information Technologies and Engineering of Aviation Computer Systems"

Topic: Intelligent control of a UAV swarm based on the leader-follower model in conditions of limited communication

Executor: higher education applicant Pavlo Shemchuk

Head: Art. Lecturer, Ph.D. Serhiy Dolhorukov

Norm controller: _____Filiashkin M.K.

(signature)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
«КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
Факультет аеронавігації, електроніки та телекомунікацій
Кафедра авіаційних комп'ютерно-інтегрованих комплексів

Освітній ступінь: бакалавр
Спеціальність 151 «Автоматизація та комп'ютерно-інтегровані технології»
Освітньо-професійна програма «Інформаційні технології та інженерія авіаційних комп'ютерних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Віктор СИНЄГЛАЗОВ
" ____ " _____2025 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи студента

Шемчука Павла Володимировича

1. Тема роботи: «Інтелектуальне керування роєм безпілотних літальних апаратів на основі моделі лідер–послідовник в умовах обмеженого зв'язку»

2. Термін виконання роботи: з 05.05.2025 р. до 10.06.2025 р.

3. Зміст пояснювальної записки (перелік питань, що підлягають розробці): 1. Аналіз сучасного стану розвитку в галузі інтелектуального управління роями БПЛА; 2. Інтелектуальне управління роями БПЛА, особливості та проблеми; 3. Розробка інтелектуальної системи управління роєм БПЛА; 4. Реалізація системи та експериментальні результати.

4. Перелік обов'язкового графічного матеріалу: 1. Структурна схема архітектури системи керування роєм БПЛА; 2. Схема зв'язку між агентами рою; 3. Схема моделі "лідер–послідовник"; 4. Діаграма логіки

ухилення від перешкод (на основі APF); 5. Візуалізація траєкторії руху БПЛА під час виконання місії; 6. Блок-схема алгоритму керування роєм.

5. Календарний план-графік:

№ п/п	Завдання	Термін виконання	Відмітка про виконання
1.	Аналіз літературних джерел	05.05.2025 – 08.05.2025	Виконано
2.	Збір інформації	09.05. 2025 – 12.05.2025	Виконано
3.	Аналіз сучасного стану розвитку в галузі інтелектуального управління роями БПЛА	13.05.2025 – 16.05.2025	Виконано
4.	Інтелектуальне управління роями БПЛА, особливості та проблеми	17.05.2025 – 22.05.2025	Виконано
5.	Розробка інтелектуальної системи управління роєм БПЛА	23.05.2025 – 31.05.2025	Виконано
6.	Реалізація системи та експериментальні результати	01.06.2025 – 07.06.2025	Виконано
7.	Висновки по роботі	08.06.2025	Виконано
8.	Оформлення пояснювальної записки	09.06.2025	Виконано
9.	Створення презентації	10.06.2025	Виконано

6. Дата видачі завдання _____

Керівник:_____Долгоруков С.О.

Завдання прийняв до виконання: _____ Шемчук П.В.

“ ____ ” _____2025 р.

STATE UNIVERSITY "KYIV AVIATION INSTITUTE"
Faculty of Air Navigation, Electronics and TelecommunicationsDepartment of Aviation Computer-Integrated Complexes

Educational degree: Bachelor Specialty 151 "Automation and Computer-Integrated Technologies"
Educational and professional program "Information Technologies and Engineering of Aviation Computer Systems"

CONFIRM
Head of the Department
_____ Viktor SYNIEHLAZOV
“ ____ ” _____2025 p.

TASKS

for the performance of the qualification work of the student

Shemchuk Pavlo

1. Topic of work: "Intelligent control of a swarm of unmanned aerial vehicles based on the leader-follower model in conditions of limited communication"

2. Term of work: from 05.05.2025 to 10.06.2025.

3. Content of the explanatory note (list of issues to be developed): 1. Analysis of the current state of development in the field of intelligent control of UAV swarms; 2. Intelligent control of UAV swarms, features and problems; 3. Development of an intelligent UAV swarm control system; 4. Implementation of the system and experimental results.

4. List of mandatory graphic material: 1. Structural diagram of the architecture of the UAV swarm control system; 2. Scheme of communication between swarm agents; 3. Scheme of the "leader-follower" model; 4. Diagram

of the logic of obstacle avoidance (based on APF); 5. Visualization of the trajectory of the UAV during the mission; 6. Block diagram of the swarm control algorithm.

5. Calendar schedule:

No. p/p	Task	Deadline	Mark of execution
1.	Analysis of literary sources	05.05.2025 – 08.05.2025	Performed
2.	Information Collection	09.05. 2025 – 12.05.2025	Performed
3.	Analysis of the current state of development in the field of intelligent control of UAV swarms	13.05.2025 – 16.05.2025	Performed
4.	Intelligent control of UAV swarms, features and problems	17.05.2025 – 22.05.2025	Performed
5.	Development of an intelligent UAV swarm control system	23.05.2025 – 31.05.2025	Performed
6.	System implementation and experimental results	01.06.2025 – 07.06.2025	Performed
7.	Conclusions on the work	08.06.2025	Performed
8.	Drawing up an explanatory note	09.06.2025	Performed
9.	Create a presentation	10.06.2025	Performed

6. Date of assignment _____

Head:_____Dolhorukov S.O.

The task was accepted for execution:_____Shemchuk P.V.

" ____ " _____2025

ABSTRACT

The explanatory note to the qualification work “Intelligent control of a swarm of unmanned aerial vehicles based on a leader-follower model in conditions of limited communication” contains: 76 pages, 11 figures, 27 sources.

Keywords: UAV swarm, intelligent control, leader-follower model, limited communication, decentralized control, ROS, Deep Q-Network, self-organization, fault tolerance, autonomous agents, trajectory optimization.

The purpose of this study is to develop an intelligent decentralized control system for a swarm of unmanned aerial vehicles based on the “leader-follower” model, which provides effective interaction between swarm elements, optimization of movement trajectories and dynamic obstacle avoidance in conditions of limited or unstable communication.

The object of the study is a control system for a swarm of unmanned aerial vehicles, which operates in a dynamic environment with a limited or unstable communication channel between individual elements of the swarm. Such a system involves the interaction of autonomous agents (UAVs), which must ensure coordinated movement, coordination of actions and achievement of a common goal even in the presence of external obstacles or loss of part of the information exchange.

The subject of the study is methods, models and algorithms for intelligent decentralized control of UAV swarms, built on the basis of the "leader-follower" principle. Special attention is paid to the use of algorithms for optimizing movement trajectories, methods for dynamic obstacle avoidance, as well as adaptation of control to changes in the structure of the swarm and the quality of communication between elements. The subject covers theoretical aspects of the development of swarm behavior models, technical implementation of interaction algorithms and simulation modeling of their operation in conditions close to real ones.

The work uses artificial intelligence methods, including deep reinforcement learning (Deep Q-Network), trajectory optimization algorithms (Whale Optimization Algorithm), sensor information processing methods, and local navigation (Artificial Potential Fields).

A software implementation of the system based on the ROS environment was developed, simulation in Gazebo was performed, and typical mission scenarios were implemented, including obstacle avoidance, search and rescue, and loss zone avoidance. The results of experiments confirming the effectiveness of the proposed architecture under limited connectivity conditions are presented.

CONTENTS

INTRODUCTION

In the modern world, unmanned aerial vehicles (UAVs) are playing an increasingly important role in various spheres of human activity, from defense and security to agriculture, logistics, environmental monitoring and emergencies. Technological developments allow us to move from the use of individual UAVs to the use of their collective groups - swarms, which are able to perform complex tasks faster, more efficiently and more safely due to the division of responsibilities and coordination of actions.

Of particular interest is the organization of a UAV swarm according to the "leader-follower" model, which allows achieving coordinated movement of a group of devices with minimal communication costs. In such a structure, one or more drones perform the function of a leader, determining the trajectory of movement, and the rest follow them according to certain behavioral algorithms. This greatly simplifies management and allows the swarm to act as a single dynamic system.

However, the implementation of such a system requires solving a number of complex problems. Among them are limitations on the communication channel between swarm elements, which may arise due to radio interference, limited range, or deliberate attempts to suppress the signal. In such conditions, the control system must provide stability, adaptability, and autonomy. Also critically important is solving the problems of optimizing trajectories and avoiding obstacles that arise in a real or changing environment.

The development of an intelligent decentralized UAV swarm control system that combines the leader-follower model, modern optimization and avoidance algorithms, and is also capable of operating in limited communication conditions is a promising area of research. Such a system has the potential to increase the effectiveness of UAV use in complex scenarios: in combat conditions, in hard-to-reach areas, during disasters or rescue operations.

Relevance

The relevance of this topic is due to the growing role of UAV swarms in the military and civilian spheres. In modern conflicts and engineering solutions, there is a clear trend towards replacing single drones with coordinated groups capable of operating autonomously in complex and dynamic conditions.

At the same time, there is a need to ensure effective control of such swarms in conditions of limited or unstable communication. In such situations, centralized systems become less effective, which necessitates the transition to decentralized solutions that ensure stable operation even in the absence of feedback.

The “leader-follower” model is promising for implementing coordinated drone movement with minimal costs for computing and data transmission. It allows you to form a flexible swarm structure with the possibility of restructuring in the event of the loss of one of the elements.

The integration of artificial intelligence algorithms, in particular trajectory optimization, obstacle avoidance, adaptive planning and reinforcement learning, is becoming particularly relevant. This makes it possible to implement autonomous and adaptive systems capable of independent decision-making in a changing environment.

Such a system has wide practical application possibilities: in search and rescue operations, in monitoring critical infrastructure facilities, in the agricultural sector, logistics, defense and many other areas.

All this indicates the high practical and scientific significance of the research devoted to the development of an intelligent UAV swarm control system based on the “leader-follower” model in conditions of limited communication.

Purpose and objectives of the study:

The purpose of this study is to develop an intelligent decentralized control system for a swarm of unmanned aerial vehicles based on the “leader-follower”

model, which provides effective interaction between swarm elements, optimization of movement trajectories and dynamic obstacle avoidance in conditions of limited or unstable communication.

To achieve this goal, the following tasks are expected to be performed:

- to analyze modern approaches to the organization of UAV swarms and decentralized control;
- to investigate the features of the “leader-follower” model in the context of its application for autonomous swarm navigation;
- to consider the problem of limited communication between drones and its impact on coordination in the system;
- to justify the choice of algorithms for trajectory optimization and dynamic avoidance suitable for conditions with limited information exchange;
- to develop the architecture of an intelligent control system that takes into account decentralization, autonomy and minimization of dependence on communication;
- create a simulation model of the system using the selected approach to swarm management;
- conduct a series of experimental studies to verify the performance and effectiveness of the proposed system under different scenarios (leader change, loss of communication, presence of obstacles);
- analyze the results obtained and formulate practical conclusions and recommendations for further improvement of the developed system.

Object and subject of study:

The object of the study is a control system for a swarm of unmanned aerial vehicles, which operates in a dynamic environment with a limited or unstable communication channel between individual elements of the swarm. Such a system involves the interaction of autonomous agents (UAVs), which must ensure

coordinated movement, coordination of actions and achievement of a common goal even in the presence of external obstacles or loss of part of the information exchange.

The subject of the study is methods, models and algorithms for intelligent decentralized control of UAV swarms, built on the basis of the "leader-follower" principle. Special attention is paid to the use of algorithms for optimizing movement trajectories, methods for dynamic obstacle avoidance, as well as adaptation of control to changes in the structure of the swarm and the quality of communication between elements. The subject covers theoretical aspects of the development of swarm behavior models, technical implementation of interaction algorithms and simulation modeling of their operation in conditions close to real ones.

CHAPTER 1
ANALYSIS OF THE CURRENT STATE OF DEVELOPMENTS IN THE FIELD OF INTELLIGENT CONTROL OF UAV SWARMS

1.1. Concept and types of UAV swarms

In the current context of the development of unmanned technologies, the concept of UAV swarms is gaining increasing importance, especially in the military, search and rescue, agricultural, environmental, and engineering and cartographic fields. Swarm technology allows for effective task performance through the interaction of a large number of drones acting as a single coordinated mechanism.

UAV swarm is a collection of unmanned aerial vehicles that are able to coordinate their actions in real time to achieve a common goal without constant intervention from the operator. Such a system imitates the behavior of natural swarms, flocks of birds or schools of fish, where each element acts autonomously, but in coordination with its neighbors.

The main characteristics of a UAV swarm include:

- distributed (lack of a single control center or its limited role);
- scalability (increasing the number of agents does not reduce work efficiency);
- flexibility (adaptation to environmental changes);
- failure tolerance (the loss of individual elements does not lead to the failure of the entire mission).

There are several classifications of UAV swarm types depending on the principles of organization, level of autonomy, methods of coordination and nature of communication between agents. Below are the main types of swarms that are most common in theory and practice.

According to the management principle:

- **Centralized swarms -** control is carried out from a single command center or from a single leading UAV. This type of swarm provides high accuracy, but

is vulnerable to central node failures and is not suitable for operation in conditions of limited or unstable communication.

- **Decentralized swarms** — each UAV makes decisions based on local information received from its neighbors or from sensors. The system is more flexible, reliable, and resilient to the loss of individual elements. This is the approach most often used in modern research on swarm systems.
- **Hierarchical swarms -** are a combination of centralized and decentralized control. The swarm has leader agents who are responsible for certain subtasks or groups. The rest of the drones execute commands, while maintaining the ability to act autonomously. This structure provides a good balance between control efficiency and adaptability.

By nature of interaction:

- **Cooperative swarms** — all participants interact with each other, exchange information, coordinate actions. Each drone is part of the collective mind, which allows it to effectively respond to changing situations. This type of swarm is most often implemented using artificial intelligence algorithms, machine learning or behavioral models.
- **Competitive or autonomous swarms -** each UAV performs its own individual task, but within the framework of a common plan. This type of interaction is often used when dividing up areas of observation or navigating in an environment with a large number of targets.
- **Hybrid models -** combine cooperative and autonomous behavior depending on the environmental conditions or the type of task. For example, drones can operate autonomously when scouting an area, but switch to cooperative mode when detecting objects of interest.

By the level of autonomy:

- **Semi-autonomous swarms -** some decisions are made by the operator, others by the system. This is a transitional option that is often used in practice.
- **Fully autonomous swarms -** the system is capable of functioning without external control, including route planning, obstacle avoidance, and traffic conflict resolution. This is the most complex type, which is actively researched in the context of robotic systems and AI.

By functional purpose:

- **Swarms for monitoring and surveillance** include small UAVs with optical or multispectral cameras. Used in ecology, agriculture, rescue operations.
- **Search and rescue swarms -** focused on rapid coverage territories, identifying victims or anomalies. Collective search algorithms are often used.
- **Combat swarms -** used in the military for reconnaissance, strikes, or interference against the enemy. They require high coordination, the ability to avoid obstacles, and protection from electronic countermeasures.
- **Transport swarms -** coordinated movement of cargo or logistics facilities.

Thus, understanding the types and structures of UAV swarms is a key step in building effective control models. The chosen architecture directly affects algorithmic solutions, system reliability, and its scalability and adaptability in a real environment.

1.2. Classification of UAV swarm control methods

Controlling a swarm of unmanned aerial vehicles (UAVs) is a complex task that requires effective information sharing, synchronization of actions and ensuring security when performing joint tasks. Depending on the application conditions, communication availability, number of agents and control accuracy requirements, there are a number of swarm control methods. These methods can be classified according to various criteria: control architecture, algorithmic basis, level of adaptability and scalability.

Classification by control architecture:

- **Centralized control** involves the presence of a single management element or central node, which distributes tasks among all agents of the swarm. The advantages are high coordination accuracy and simplified implementation of control algorithms. At the same time, this approach is vulnerable to failures and communication disruptions: loss or overload of the central element leads to degradation or complete shutdown of the entire system. Such models are most often used in simulations or under conditions of guaranteed communication (for example, in laboratory conditions).
- **Decentralized control** is based on autonomous adoption decisions by each drone based on local information (neighbor positions, sensor data, navigation signals). Such a system does not have a single center, which increases its resilience to failures, allows it to work in conditions of limited or unstable communication, and provides flexibility and scalability. The main challenge of decentralized models is the need for effective coordination algorithms at the local level that ensure global consistency.
- **Hierarchical control** combines elements of centralization and decentralization. Some UAVs act as leaders or coordinators (e.g., one or more drones with higher computing power or more precise navigation), while other drones act as followers, following the leaders' commands. This allows you to increase the controllability of the system while maintaining partial autonomy.

Classification by algorithmic approach:

- **The rule-based model** is based on simple local instructions that determine the behavior of each agent. Typical rules include:
    - maintaining distance from neighbors;
    - orientation in the direction of movement of neighbors;
    - following the center of mass of the group;

- avoiding obstacles.

An example is the Boids algorithm developed by Craig Reynolds, which simulates the behavior of flocks of birds. Such approaches have low computational costs and scale well, but often do not guarantee optimality or stability in complex environments.

- **Methods of potential fields** — involve mathematical modeling of the environment as a force field, where the target attracts the agent, and obstacles and other agents repel. The agent moves under the influence of the net force. This approach is simple to implement, but has a tendency to get stuck in local minima, especially in a complex environment.
- **The leader-follower model** — one or more agents determine the trajectory of movement, and the others follow them, observing the distance, direction and speed. This approach is well suited for swarms with a limited communication channel, allows to reduce the amount of information exchange and improves the structure of the system. It is possible to implement both with fixed leaders and with dynamic assignment of the leader based on context (e.g., energy reserves, GPS accuracy, availability of communication).
- **Agent-based** modeling treats each drone as an intelligent agent with its own behavior, goals, and ability to interact. Such models are flexible, but require complex implementation and large computational resources.
- **Artificial Intelligence and Machine Learning** — are used to learn behavioral models in complex dynamic environments. For example, reinforcement learning allows agents to optimize their behavior through repeated interactions with the environment. AI can be used in both centralized and decentralized architectures.
- **Optimization algorithms** are used to find optimal trajectories, minimize energy consumption, and optimally place agents. Popular ones are:

- Particle Swarm Optimization (PSO) algorithm;
- genetic algorithms;
- Ant Colony Optimization (ACO) algorithm;
- tabu search algorithms.

By the level of adaptivity:

- **Agent-based modeling** treats each drone as an intelligent agent with its own behavior, goals, and ability to interact. Such models are flexible, but require complex implementation and large computational resources.
- **Adaptive systems** change their behavior depending on changes environment, communication availability, role distribution in the swarm, or the occurrence of obstacles. Such systems are more flexible, fault-tolerant, and better suited to real-world application conditions.

1.3. The leader-follower model: principles, advantages and disadvantages

One of the most common and effective models for coordinating unmanned aerial vehicles in a swarm is the “leader-follower” model. It is based on one or more vehicles acting as leaders, setting the basic parameters of movement, while the others follow them, maintaining relative formation, speed, and direction.

Principles of the model

In this model, each follower receives information about the position and trajectory of the leader (or the nearest agent in the case of extended variations) and uses local sensors or communication channels to maintain formation. The main principles of implementation are:

- determination of the leader's role (dynamically or statically);
- maintaining a constant distance and orientation relative to the leader;
- adaptation to changes in the leader's position in real time;
- the ability to reorganize the swarm structure when the leader is lost.

The model can be implemented in both centralized (with one main leader) and decentralized (several local leaders or role exchange between agents) forms.

Advantages of the model:

- **Ease of implementation.** Algorithmically, the model is less complex than fully decentralized methods.
- **Scalability.** It is easy to scale the swarm by adding new followers without significantly affecting the system.
- **Controllability.** The presence of a leader ensures a clear direction and dynamics of the group's movement.
- **Reduced computational load.** Followers do not require global planning — they just follow the leader.

Disadvantages of the model:

- **High dependence on the leader.** The loss or refusal of the leader can disrupt the swarm.
- **Limitations in complex environments.** The model is less effective in highly dynamic conditions or when there are many obstacles.
- **Communication difficulties.** In the event of limited communication or delays in data transmission, synchronization is disrupted.
- **Less flexibility.** Compared to decentralized swarms, leader-follower swarms are less adaptable to unpredictable situations.

1.4. Communication problems in swarms: limitations, delays, losses

Communication between UAV swarm agents is critical for effective coordination and control. It is data exchange that ensures smooth movement, obstacle avoidance, adaptation to changes in the environment, and maintenance of the desired formation. However, in real-world applications, communication systems face a number of challenges that can significantly reduce the effectiveness of a swarm or even lead to its disorganization.

Bandwidth limitations

One of the main factors is the limited bandwidth of communication channels. In many cases, swarms use wireless protocols (Wi-Fi, ZigBee, LoRa, 4G/5G), which have physical and technical limits on the amount of information that can be transmitted, especially when there are a large number of devices in the swarm. This can lead to channel overload, packet loss, or slow status updates.

Data transmission delays

Delays are a typical phenomenon in networks with mobile nodes. Increased distance between UAVs, changing environmental conditions (obstacles, noise, interference), and poor-quality transmission routes can cause latency, which critically affects the coordination of actions in a swarm.

In particular, a delayed response to a change in the position of the leader or neighbors can lead to a break in formation.

Loss of communication

In complex terrain, urban areas, or combat conditions, complete or partial loss of communication between agents is possible. This can be the result of:

- radio jamming or attacks (e.g., using jammers);
- an agent flying outside the coverage area;
- hardware or power failures of the communication module.

In such cases, it is necessary to implement autonomous behavior algorithms that will ensure temporary independence of drones or the ability to automatically reconfigure the network.

Network topology requirements

The type of network topology (mesh, star, tree) determines the system's resistance to communication losses. In centralized systems (star-type), the loss of the central node (leader) can lead to network collapse, while in decentralized topologies (mesh), routing can be reconfigured.

Impact on control algorithms

Communication failures directly affect coordination algorithms — from forming a shared view of the situation to calculating a collective trajectory. In such conditions, it is necessary to consider:

- asynchronous updates;
- partial lack of information;
- the need to make decisions based on local data.

The development of intelligent control systems must take these limitations into account by applying approaches that increase resilience to communication losses, in particular local avoidance algorithms, built-in autonomy, leader rotation algorithms, data buffering, and data retransmission.

Thus, effective swarm control of UAVs requires consideration of all factors related to communication problems, which is particularly relevant in real-time tasks and dynamic environments.

1.5. Overview of modern swarm control and coordination algorithms

The effective functioning of a swarm of unmanned aerial vehicles (UAVs) largely depends on the approach chosen for control and coordination between agents. In conditions of limited communication, a dynamic environment, and the need for autonomous interaction, the role of intelligent algorithms capable of ensuring the adaptability, coordination, and stability of the swarm is growing.

Classification of control algorithms

Modern methods of UAV coordination are conventionally divided into the following groups:

- **Behavior-based —** models based on simple local rules (e.g., Reynolds algorithm: alignment, collision avoidance, attraction to the center of the swarm).

- **Artificial Potential Fields (APF) —** formation of virtual forces that direct UAVs in the desired direction while repelling obstacles or other agents.
- **Graph-based algorithms —** use the topology of connections between agents to build a route and coordinate actions (e.g., adjacency graphs, spanning trees).
- **Optimization methods —** the use of evolutionary or heuristic algorithms to find the best trajectories (Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Whale Optimization Algorithm (WOA), etc.).
- **Reinforcement learning —** drones learn to make decisions by interacting with the environment. The most popular are Deep Q-Learning (DQN), Proximal Policy Optimization (PPO), and Actor-Critic models.
- **Consensus-based methods —** ensure coordination of the swarm's state (position, direction, speed) by exchanging local information between neighboring agents.
- **Leader-follower models —** one or more agents set benchmarks for the rest of the swarm, which then follows their actions.

Modern approaches in conditions of limited connectivity

In environments with limited or unstable connectivity, the following are used:

- **Local coordination models** that do not require a complete data exchange network.
- **Decentralized control methods** that allow each agent to make decisions independently.
- **Hierarchical models**, where the role of leader can be transferred or updated in real time.
- **Communication recovery algorithms** through signal relay or the formation of temporary communication bridges.

Examples of application:

- **Deep Reinforcement Learning:** used for complex swarm maneuvers in dynamic environments (in particular DQN, DDPG).
- **Swarm Intelligence:** ACO and PSO are widely used for distributed route planning.
- **Artificial Potential Fields:** effective in real-time obstacle avoidance and formation tasks.
- **Flocking-based control:** popular in small drones (e.g., Crazyflie) for coordinated indoor flight.

Modern research increasingly combines several methods to create hybrid systems (e.g., APF + RL or PSO + consensus), which allows for a combination of flexibility, adaptability, and control accuracy.

1.6. Justification of intelligent control of UAV swarms

Controlling swarms of unmanned aerial vehicles (UAVs) is a complex task that requires adaptive, reliable, and scalable solutions. Classic control approaches—such as rigid rules, centralized schemes, or methods with predefined behavior logic—show limited effectiveness in complex or dynamic environments. This is particularly noticeable in situations with limited connectivity, changing environments, or high requirements for system autonomy. Therefore, modern research is actively introducing intelligent control approaches, including machine learning methods, optimization algorithms, and hybrid interaction models.

One of the key problems of classical systems is their low flexibility. In the event of unpredictable changes, such as the emergence of new obstacles, the loss of a swarm element, or a loss of communication with the central node, rigid algorithms are often unable to effectively reorganize behavior. In contrast, intelligent algorithms, especially those based on reinforcement learning (RL), allow agents to autonomously form behaviors that maximize goal achievement even in complex and partially uncertain environments.

Another important factor is scalability. In centralized systems, as the number of drones increases, so does the computational load and the need for extensive data exchange, making real-time operation impossible. In decentralized intelligent systems, each agent makes decisions locally, allowing for distributed computation and increased fault tolerance.

Intelligent methods also provide adaptability to new scenarios. For example, trajectory optimization algorithms such as the **Whale Optimization Algorithm (WOA) or Particle Swarm Optimization (PSO)** allow optimal routes to be formed taking into account the objective function, energy consumption, and obstacle configuration. In combination with potential fields, they form hybrid models that avoid local minimum traps and maintain formation.

Importantly, intelligent systems can operate with partial or complete loss of communication, relying on local sensors and a pre-trained behavior model. This ability to make autonomous decisions is critical in combat, emergency situations, or operations in highly obstructed environments.

1.7. Review of current research in the field of UAV swarm control

The development of unmanned aerial vehicles (UAVs) has opened up new opportunities for the implementation of complex tasks that require the simultaneous operation of a large number of agents. In this regard, research related to swarm control systems has become widespread in the last decade. UAVs operating in swarms have advantages in terms of flexibility, fault tolerance, and scalability, but require effective algorithms for coordination, motion planning, and obstacle avoidance.

Among the classical approaches to coordination, algorithms based on local behavior rules remain the most common. Publication [2] provides a detailed analysis of a behavioral model in which each drone follows simple rules for alignment,

repulsion, and attraction to the center of the swarm. Such models have high computational efficiency but can be unstable in complex or dynamic environments.

The leader-follower model plays an important role in swarm formation, providing a hierarchical structure for interaction. Studies [12], [16] show that combining this model with local optimization methods allows for high formation coherence even in the event of partial loss of communication with the leader. In [10], a variant of adaptive interaction is demonstrated, where followers orient themselves not only to the leader but also to data from neighbors, reducing dependence on centralized information.

Potential field methods are actively used for evasion and maneuvering tasks in environments with obstacles. Article [15] proposes an extended version of the classic APF (Artificial Potential Fields), which takes into account both static and dynamic objects in the drone's area of operation. The method demonstrates good results in real-time modes, especially when used in combination with local learning algorithms.

Along with classical methods, heuristic and metaheuristic optimization algorithms are increasingly being used. For example, publication [14] considers the application of the Whale Optimization Algorithm (WOA) for constructing motion trajectories under conditions of limited information. Similarly, [23] demonstrates the effectiveness of using particle swarm optimization (PSO) algorithms and genetic strategies to find a globally optimal path while minimizing energy consumption.

The scientific community pays particular attention to machine learning algorithms, in particular reinforcement learning. In [8] and [10], systems based on Deep Q-Network are developed, where the swarm leader self-learns the optimal route by responding to changes in the environment and the actions of other agents. Such models are well suited for adaptive scenarios where it is impossible to predict all obstacle configurations or mission changes in advance.

An equally important research topic is ensuring stable communication between agents. In [7] and [19], we provide an overview of network topological structures that allow information exchange to be maintained in the event of partial communication failures. Scenarios involving spoofing or signal jamming are also relevant. [22] and [18] describe decentralized coordination methods that allow a swarm to adapt to the loss of a communication channel without compromising the target function.

A number of works (e.g., [3], [4], [6]) consider the physical modeling of swarms, taking into account inertial characteristics, aerodynamic forces, and speed and control constraints. Such studies bring theoretical models closer to practical application, for example, in search and rescue operations, patrolling, or autonomous delivery.

Hybrid models that combine several approaches deserve special attention: decentralized control, local behavior rules, optimization algorithms, and neural network modules. In [13], a system is presented that dynamically switches between modes according to the current situation in the environment. This ensures adaptability, autonomy, and resistance to external influences, which is especially important for real-time multi-drone platforms.

CHAPTER 2
PROBLEM STATEMENT AND SELECTION OF IMPLEMENTATION

2.1. Problem Statement of the Intelligent Swarm Control

Intelligent control of unmanned aerial vehicle (UAV) swarms is a complex multifactorial task that encompasses both technical and algorithmic aspects. In general terms, it involves creating a system that provides autonomous coordination of multiple agents performing a joint mission in the absence of centralized control.

Swarm system control involves the coordinated operation of agents with minimal information exchange. This is particularly relevant in situations where the communication channel is limited in terms of range, bandwidth, or stability. The task is complicated by the need to take into account changing environmental conditions, the dynamics of each UAV, as well as limitations on speed, energy consumption, and interaction topology.

The objective function of such a system is usually to minimize the total task execution time, maintain formation, and ensure reliable avoidance of obstacles encountered along the route. At the same time, the entire system must be resistant to partial communication or agent losses, adaptable, and scalable.

The main objective of the work is to develop an algorithmic and software base for ensuring autonomous coordinated movement of a group of UAVs, which must perform assigned missions while maintaining a stable formation, avoiding collisions with obstacles, and ensuring fault tolerance in case of loss of individual system elements.

To achieve this goal, a set of interrelated scientific and technical tasks must be solved:

- Development of a swarm architecture that will ensure decentralized interaction of agents with a minimum amount of data exchange, including mechanisms for reassigning roles in case of leader loss.

- Creation of a strategic route planning mechanism at the leader level that will allow the swarm's trajectory to be adapted to changes in the environment.
- Development of local obstacle avoidance algorithms for each UAV based on the artificial potential field method, which will ensure autonomous decision-making in conditions of limited communication.

In mathematical terms, the problem boils down to optimizing trajectories and control signals for each swarm element, taking into account a set of constraints: safety distances between agents, platform dynamics, communication links between elements, and shared resources (e.g., energy). The algorithm must be implemented in such a way that, in the event of loss of access to global information, each agent can continue the mission using only local observations or data from its closest neighbors.

Formally, the optimization problem can be written as:

$$\min_{u_i(t)} J = \sum_{i=1}^{N} \int_{t_0}^{t_f} \left(\alpha \left\| x_i(t) - x_{\text{target}}(t) \right\|^2 + \beta \left\| u_i(t) \right\|^2 \right) dt$$

provided that: $\left\| x_i(t) - x_j(t) \right\| \geq d_{min};\ x_i(t+1) = f\left(x_i(t), u_i(t)\right)$

where:

- $x_i(t)$ state of agent i at time t;
- $u_i(t)$ control input for agent i;
- $x_{target}(t)$ desired position/formation;
- d_{min} minimal state distance between agents;
- α, β weighting coefficients for accuracy and energy consumption.

Thus, statement of the problem of intelligent swarm control is a combination of requirements for accuracy, stability, adaptability, and efficiency of an autonomous multi-agent system under conditions of limited communication.

Therefore, the expected result of the work is the creation of an autonomous swarm control system that will be scalable, flexible, and capable of functioning in conditions of limited communication infrastructure.

2.2. Overview of unmanned aerial vehicle swarm motion models

Unmanned aerial vehicle swarms are complex multi-component systems in which each drone performs tasks as part of a joint mission. To effectively control such swarms, it is necessary to develop mathematical motion models that allow describing and predicting their behavior under various conditions, such as communication constraints, obstacles, and interactions between drones. This review discusses the main UAV swarm motion models used in current research and practical applications.

Leader-follower model

The leader-follower model is one of the simplest and most widely used models for swarm control of UAVs. In this model, one drone acts as the leader, setting the flight path for the entire swarm, while the other drones, acting as followers, try to follow the leader while maintaining a certain distance or orientation.

The advantages of this model are its simplicity and clear organization of swarm movement, which is useful for tasks requiring precision and coordination. However, the main disadvantage is the limitation in case of loss of communication with the leader or technical failure of one of the drones, which can lead to a disruption of the swarm's stability.

Application: The leader-follower model is effectively used in search and rescue tasks, as well as in tasks where it is important to maintain a stable formation with a small number of drones.

Distributed models (distributed control)

In distributed control models, a swarm of UAVs does not have a centralized leader. Each drone in the swarm interacts only with its closest neighbors, choosing

actions based on local information. This approach significantly increases the flexibility of the swarm and ensures its resilience to failures.

The advantages of this model include a high level of adaptability and fault tolerance, as each drone can make decisions independently, based solely on its interactions with other drones. However, the disadvantage is the complexity of configuring the algorithms for interaction between drones, as the model requires coordinated work without centralized control.

Application: Distributed models are used in autonomous drone control systems where high fault tolerance and independent decision-making are required, such as in agriculture, environmental monitoring, or military applications.

Swarm self-organization

Swarm self-organization models are based on the principle that drones independently coordinate their actions using local information, without the need to receive centralized commands. Each agent (drone) assesses its place in the system and makes decisions based on the current state of the environment and the behavior of other agents.

Advantages of this model include flexibility and the ability of the swarm to adapt to changing conditions. A disadvantage is the difficulty of ensuring stability in highly complex environments or when there are a large number of drones that can interact with each other.

Application: Swarm self-organization models are used in complex or unpredictable environments, such as search and rescue operations or military missions, where each drone must act autonomously based on local information.

Models with evolutionary algorithms and genetic methods

Models with evolutionary algorithms and genetic methods use approaches inspired by natural evolutionary processes to optimize swarm movement strategies.

Genetic algorithms or evolutionary methods allow drones to gradually adapt their strategies through selection, crossover, and mutation.

The advantages of this approach lie in its ability to find optimal solutions for complex problems with many variables. However, the disadvantages are high computational complexity and long training times.

Application: These models are used for optimization problems where it is necessary to find the most efficient movement strategy for a swarm, for example, in mission planning or selecting optimal routes for drones.

Neural network-based models (DRL)

Deep reinforcement learning (DRL) models use neural networks to optimize the behavior of drones in a swarm. The agent learns to select the most beneficial actions to maximize its reward in the environment using deep learning methods to process large amounts of data.

The advantages of this approach are its ability to adapt to complex, unpredictable environments and find optimal solutions based on experience. However, the disadvantages are the high complexity of implementation and the need for significant computing resources for training.

Application: DRL models are used in real-world tasks such as autonomous vehicle control, robotics, and complex gaming environments where decisions must be made under resource constraints.

2.3. Justification for choosing the “leader-follower” model

In the context of building a UAV swarm control system, an important step is choosing a model of interaction between agents. Among the available approaches, such as centralized control, complete decentralization, hierarchical models, or topology-oriented strategies, the leader-follower model occupies a special place due to its simplicity, scalability, and technical feasibility.

This model assumes that one or more agents perform the functions of a leader, setting the trajectory or behavioral guidelines for other members of the swarm — followers. Followers do not receive complete information about the global state of the system, but rely only on data received from the leader or their closest neighbors. This approach reduces the load on the communication channel and increases the system's resistance to failures.

Roles in the leader-follower model:

Leader: an agent that determines the overall trajectory of the swarm, makes strategic decisions, and transmits information to followers. The leader may have access to global information about the environment and mission objectives.

Follower: an agent that follows the leader, maintaining a specified distance and orientation. Followers use local sensors and information from the leader to adjust their movement.

Mathematical model of tracking:

Each follower agent i changes its position according to the position of the leader or neighbor j, using a simple rule:

$$\dot{x}_i(t) = k_p \left(x_j(t) - x_i(t) \right)$$

where:

- $x_i(t), x_j(t)$ the position of agent i and leader (or neighbor) j at time t;
- $\dot{x}_i(t)$ the rate of change in the position of agent i;
- k_p the proportionality coefficient (tracking parameter).

In vector form for 2D/3D, this can also be written as:

$$v_i = k_p \cdot \left(x_j - x_i \right)$$

This rule implements local control: each agent aligns itself relative to the leader or neighbor without requiring global information. It can be extended to several neighbors using a weighted average.

The Figure 2.1 illustrates how the leader transmits information to followers, who in turn adjust their movement according to the data received. This structure allows the swarm to adapt effectively to changes in the environment and perform complex tasks with minimal resource consumption.

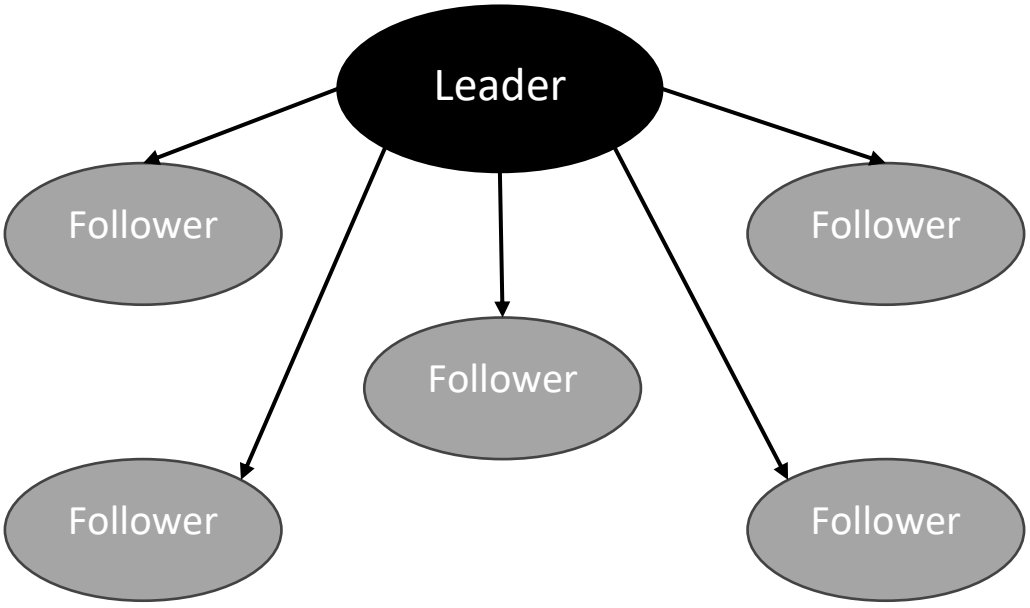

Fig. 2.1. Interaction diagram in the “leader–follower” model

In the selected architecture, this model is used as the basis for building the behavior logic of a swarm in conditions of limited communication. In particular, the leader acts as a trajectory generator and the main sensor node, while each follower maintains a certain distance and orientation relative to it or other neighbors. In case of communication loss or failure of the leader, the system switches to local control mode with partial formation preservation or selection of a new leader.

Thus, the choice of the leader-follower model is driven by the desire to balance simplicity of implementation and the ability of the system to operate autonomously in a real environment with limited communication resources.

2.4. Methodology for implementing decentralization

In modern UAV swarm control systems, decentralization is considered a key principle that ensures autonomy, fault tolerance, and flexibility in responding to

changes in the environment. Unlike centralized schemes, where all decisions are made by a single control center, decentralized models assume that each agent makes decisions locally, relying on its own sensors or limited information from neighbors.

In this work, decentralization is implemented within a leader-follower model through the organization of local interactions. Each follower does not rely on complete information about the global state of the swarm, but follows the leader or the nearest neighbors according to predefined rules: maintaining optimal distance, speed alignment, and obstacle avoidance. This approach minimizes the need for extensive radio communication and ensures system scalability.

For practical implementation, a "k-neighbor" communication topology is used (for example, each agent exchanges information only with its three closest neighbors). Data transmission is not continuous, but event-driven or limited in frequency, which significantly reduces the load on the channel. If communication with the leader is lost, the agent switches to local control, maintaining its position in the formation or forming a new structure with other neighbors.

In a decentralized model, each agent i receives information only from a limited number of k closest neighbors belonging to the set N_j. Its control action is formed as a combination of local rules: maintaining distance, speed alignment, and movement toward the target. This can be written as:

$$u_i(t) = w_1 u_{allign,i}(t) + w_2 u_{coh,i}(t) + w_3 u_{sep,i}(t)$$

where:

- $u_{allign,i}(t)$ alignment vector (within the velocities of neighbors);
- $u_{coh,i}(t)$ cohesion (movement toward the center of mass of neighbors);
- $u_{sep,i}(t)$ separation (repulsion from agents that are too close);
- $w_1, w_1, w_1 \in \mathbb{R}$ weights that set the priority of each component.

For example, cohesion can be described as follows:

$$u_{coh,i}(t) = \frac{1}{|N_i|} \sum_{j \in N_i} x_j(t) - x_i(t)$$

and alignment:

$$u_{allign,i}(t) = \frac{1}{|N_i|} \sum_{j \in N_i} v_j(t) - v_i(t)$$

Thus, each agent dynamically adapts its behavior according to the actions of its neighbors without the need for centralized control or global knowledge of the situation.

The Figure 2.2 shows the interaction of swarm elements in a decentralized environment: the leader sets the direction of movement, while each follower makes decisions based on local information from neighboring agents. The relationships between agents are indicated by data exchange arrows, which do not require a global controller.

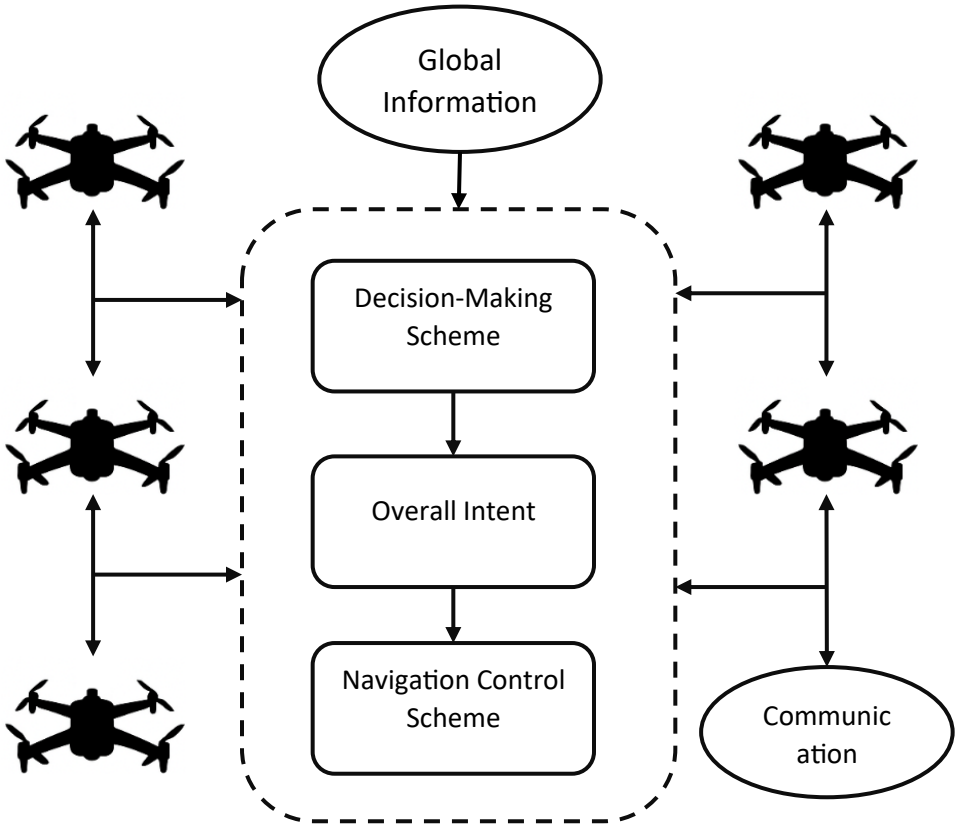

Fig. 2.2. Diagram of decentralized swarm control of UAVs

Decentralization also involves the independent generation of control signals by each agent based on local information about relative position, speed, and orientation. These signals are formed using built-in algorithms, such as potential field functions, behavioral rules, or deep learning modules. All these components ensure the flexibility and adaptability of the system to unforeseen situations in flight.

Thus, the developed decentralization methodology allows to increase the reliability of the swarm in conditions of limited communication, reduce dependence on the central node, and improve the system's resistance to partial failures or information losses.

2.5. Consideration of communication restrictions

One of the main features of swarm systems is their high dependence on inter-agent communication. However, in real-world conditions, communication between unmanned aerial vehicles is subject to significant limitations, which may be caused by physical factors, intentional interference, or technical characteristics of the vehicles. Therefore, taking these limitations into account is a critical element in the design of a swarm control system.

Limitations can be of various nature. For example, the range of transceivers determines the maximum distance at which agents can exchange information. In complex or urban environments, communication quality is affected by multiple signal reflections, building shielding, changes in channel frequency characteristics, and interference. In military or special conditions, deliberate attacks on the network (jamming, spoofing) may occur, further complicating communication.

In addition, channel bandwidth limitations, energy consumption for data transmission, and communication delays must be taken into account. Even in stable communications, latency can lead to a loss of swarm coordination, especially when configurations change rapidly or in response to obstacles.

In the control model used in this work, communication is implemented in a limited mode: each agent interacts only with a certain number of nearest neighbors using simple status signals (e.g., position, speed, trajectory command). This reduces dependence on the global communication channel and allows the system to adapt to situations where communication with part of the swarm or the central base station is completely lost.

In the implemented model, each agent can exchange information only with those agents that are within a certain radius R. This can be described by the condition:

$$j \in N_i(t) \leftrightarrow \left\| x_j(t) - x_i(t) \right\| \leq R$$

where:

- $N_i(t)$ the set of neighbors of agent i at time t;
- $x_i(t), x_j(t)$ the coordinates of the corresponding agents;
- R the fixed communication radius.

The result is a dynamic communication topology that changes depending on the swarm configuration, and this is taken into account during state exchange.

Figure 2.3 illustrates the principle of communication organization in a swarm of UAVs under conditions of limited range of receiving and transmitting devices. In the presented topology, each agent (network node) has a communication channel only with a limited number of nearest neighbors, which forms local communication clusters. This structure avoids overloading the common channel, reduces the impact of delays and failures in the network, and increases the system's resistance to communication losses with individual elements. This approach is typical for decentralized swarm systems and provides a basis for autonomous decision-making by each UAV in conditions of partial or complete loss of global control.

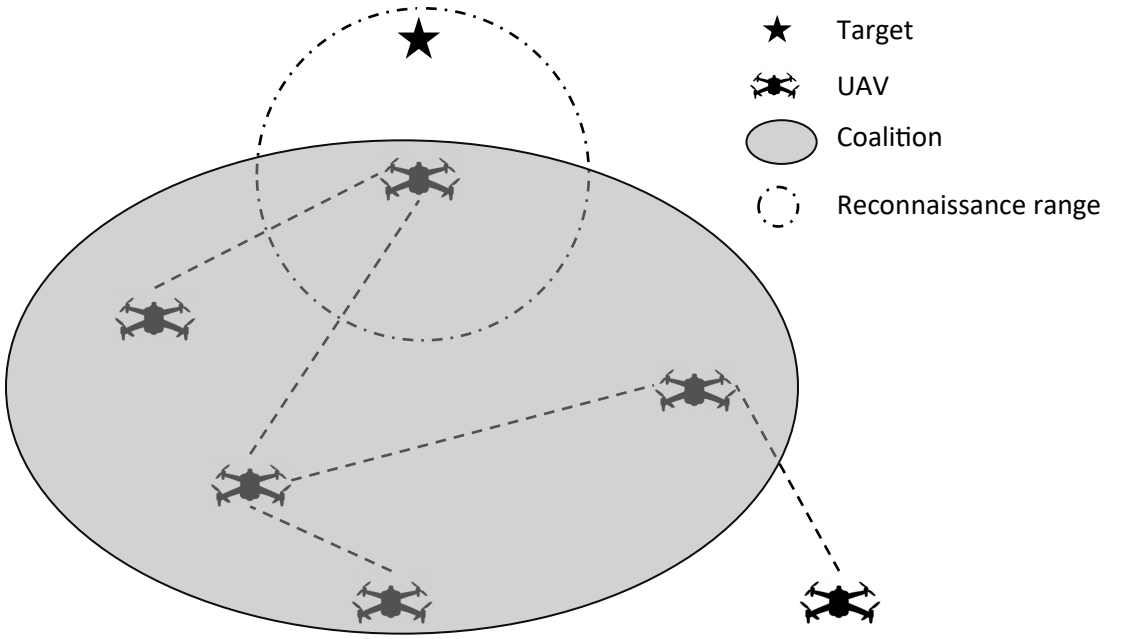

Fig. 2.3. Topology of a local communication network for a swarm of UAVs with limited range

2.6. Trajectory optimization algorithms

Building an optimal trajectory for a swarm of UAVs is a critical component of the control system, especially in conditions of limited communication, changing environments, and limited resources. The main goal is to form movement routes that would ensure the achievement of the mission objective with minimal energy consumption, avoidance of collisions, and preservation of formation.

Within the scope of this work, the **deep reinforcement learning method (Deep Q-Network, DQN)** is used to optimize the leader's trajectory. This approach combines the traditional Q-function, which evaluates the feasibility of performing a certain action in a given state, with a neural network that allows its value to be approximated in conditions of a large number of states and limited information. This

approach is particularly effective in dynamic environments where an agent learns to make decisions that maximize the total reward over a long period of time.

The basic value function (Q-function) approximated by DQN is defined as:

$$Q(s,a) = \mathbb{E}\left[r_t + \gamma \cdot \max_{a'} Q(s_{t+1}, a') \mid s_t = s, a_t = a\right]$$

where:

- s agent state (e.g., coordinates, speed, position of targets and obstacles);
- a action (control command);
- r_t reward for action;
- γ discount factor (takes into account the importance of future rewards).

During training, the neural network minimizes the loss function:

$$\mathcal{L}(\theta) = \left(r_t + \gamma \cdot \max_{a'} Q(s_{t+1}, a'; \theta^-) - Q(s_t, a_t; \theta)\right)^2$$

where:

- θ parameters of the current neural network;
- θ^- target network parameters.

The figure 2.4 shows the general architecture of UAV trajectory optimization using Deep Q-Network. The agent interacts with the environment, observes the state, selects an action based on the policy defined by the network, performs the action, receives a reward, and uses it to update the Q-value function step by step. This cycle allows the agent to learn to make effective route decisions on its own, even in complex conditions.

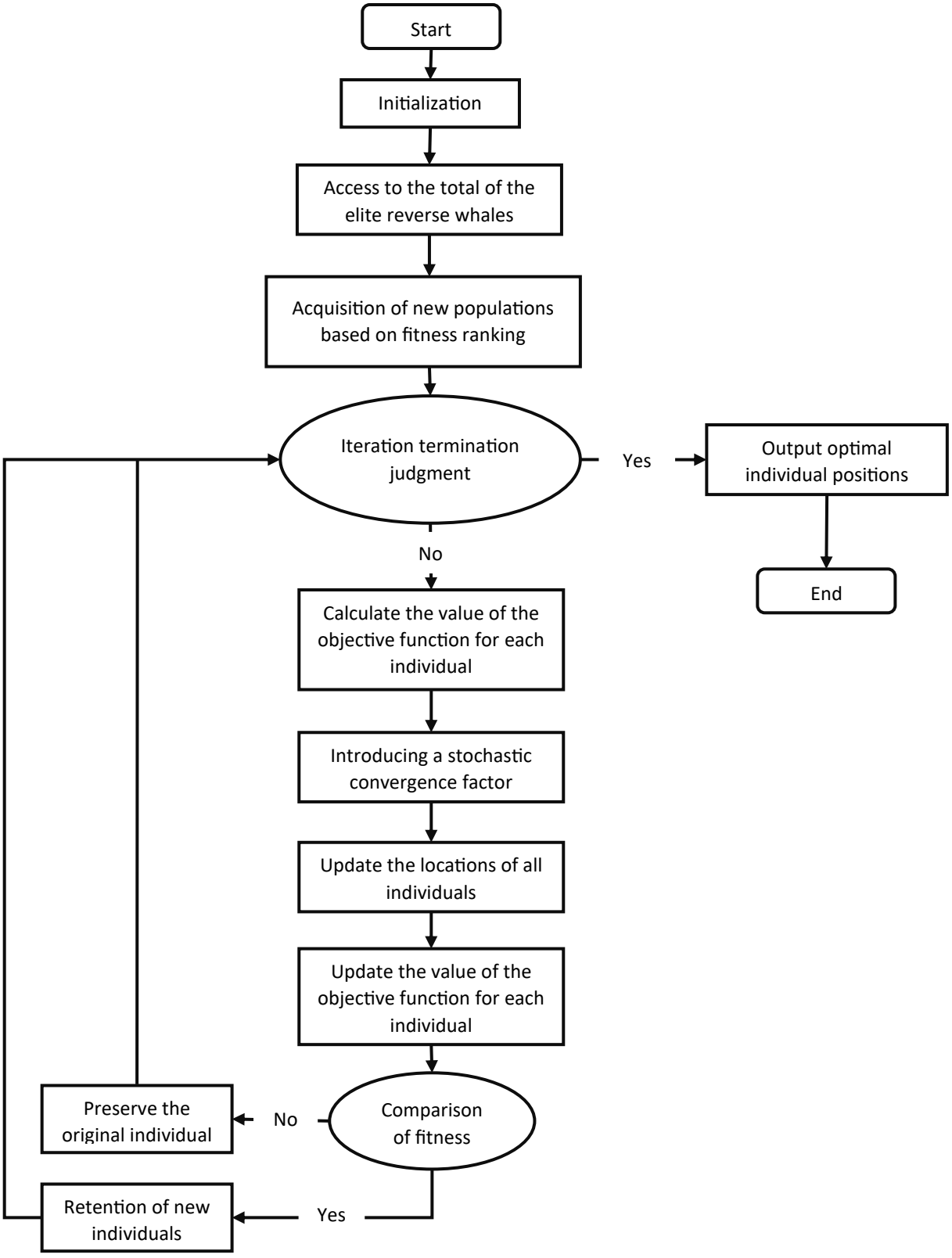

Fig. 2.4. Architecture of the DQN algorithm for trajectory optimization

During training, the agent (leader) simulates interaction with the environment, receiving feedback in the form of numerical rewards for each action. This allows it to gradually form a strategic policy — a set of actions that provide the best result in different scenarios. The input data for the algorithm are the coordinates of the target, the location of obstacles and neighbors, the speed and direction of movement, and the current state of the navigation space. The output is the selection of the direction and magnitude of the control action to transition to the next state.

In a swarm structure, this strategy allows the leader to adaptively build the optimal trajectory, avoid dynamic obstacles, and respond to communication loss while avoiding overloading the data exchange network. Followers in this system implement local rules for following the leader and also use their own position correction algorithms, which ensures a flexible and stable formation.

An auxiliary tool for preliminary route formation is the Whale Optimization Algorithm (WOA), which is used for global trajectory optimization in cases where the initial and final positions, as well as the environment configuration, are known. WOA provides an effective starting route for further refinement using DQN in real time.

Updating an agent's position in WOA is based on a mathematical model of convergence to prey:

$$\vec{X}(t+1) = \vec{X}^*(t) - A \cdot |\vec{C} \cdot \vec{X}^*(t) - \vec{X}(t)|$$

$$A = 2ar - a; C = 2r$$

where:

- $\vec{X}(t)$ the agent's position at step t;
- $\vec{X}^*(t)$ the best known position (e.g., the minimum of the objective function);
- a parameter that decreases over time;
- $r \in [0,1]$ random variable.

Thus, the use of a hybrid scheme based on DQN and WOA allows combining the advantages of global planning and local adaptation, which meets the requirements of a modern intelligent UAV swarm control system.

2.7. Real-time obstacle avoidance algorithms

Reliable real-time obstacle avoidance is one of the key tasks for ensuring the safety and stability of UAV swarm flight. This is particularly important in dynamic environments where obstacles can arise unpredictably, as well as in cases of partial or complete loss of communication between agents.

To solve this problem, the designed system uses a combination of the classic approach based on **Artificial Potential Fields (APF)** and local avoidance mechanisms implemented directly at the level of each agent. The APF method consists of creating an imaginary field where the target attracts the agent and obstacles repel it. The vector sum of these forces determines the direction of movement at the current moment in time.

In the classical approach, potential fields are formed using the following functions:

- Attractive potential energy to the target:

$$U_{att}(x) = \frac{1}{2} k_{att} \left\| x - x_{goal} \right\|^2$$

- Repulsive potential energy from an obstacle:

$$U_{rep}(x) = \frac{1}{2} k_{rep} \left(\frac{1}{\|x - x_{obs}\|} - \frac{1}{d_0} \right)^2$$

where:

- x current position of the agent;
- x_{goal} coordinates of the target;
- x_{obs} position of the obstacle;
- k_{att}, k_{rep} intensity coefficients;

- d_0 threshold distance of the obstacle.

The total force is determined as:

$$F(x) = -\nabla U(x) = -\nabla(U_{att} + U_{rep})$$

This allows you to determine the direction and magnitude of the agent's position change at any given moment, responding to changes in the environment.

The advantage of APF lies in its simplicity of implementation, high speed, and lack of need for centralized computing. Each agent independently forms a repulsive field around an object based on local sensor data or received obstacle coordinates and selects a new direction of movement. This approach integrates well with the leader-follower model logic and allows the formation to be maintained even during active maneuvering.

In environments with high obstacle density or dynamic changes, APF is supplemented with heuristic rules, including: shifting in the direction of the lowest local density, delaying maneuvers, or even temporarily leaving the formation with subsequent automatic return. Reinforcement Learning-based methods can also be used, where the agent learns to avoid obstacles without violating the specified movement strategy.

Within this system, avoidance is performed in real time, with a control decision update frequency compatible with the dynamics of the vehicle. Priority is given to speed, which is especially important in conditions of limited communication: even with complete loss of access to the leader, each agent is able to independently avoid the threat of collision while maintaining functionality and coordination.

Illustration of a potential field in the case of obstacle avoidance prediction. The UAV is represented by a dark empty square, and the obstacle is represented by a dark blue color. Cells colored green are not subject to any augmentation. Cells

subject to potential increase are represented by a gradation from brown (lowest) to red (highest).

Figure 2.5 demonstrates the basic principle of the artificial potential field (APF) method for obstacle avoidance tasks by unmanned aerial vehicles (UAVs). In the center of the scene is a drone moving toward a target while simultaneously responding to the presence of an obstacle.

Fig. 2.5. Illustration of an artificial potential field for obstacle avoidance

The scenario is that the target forms an attractive field that directs the drone in the desired direction, while the obstacle creates a repulsive field that forces the agent to deviate from a straight trajectory to avoid collision. The total force vector determines the instantaneous direction of the vehicle's motion.

It is clear that when a collision is predicted, the agent begins to maneuver around the obstacle, following the most “energy-efficient” path — that is, the one where the total action of the attractive and repulsive vectors allows it to approach the target as quickly as possible without violating safety conditions.

This model allows the drone to operate autonomously without the need for centralized trajectory calculation or constant inter-agent communication.

Importantly, the algorithm can operate in real time, taking into account only local data from sensors or onboard processing of the surrounding map.

This solution is effective not only for individual UAVs, but also for swarm structures: each agent makes decisions independently, ensuring fault tolerance, behavioral flexibility, and formation preservation, even in the absence of communication with the leader or other agents.

CHAPTER 3
DEVELOPMENT OF AN INTELLIGENT UAV SWARM CONTROL SYSTEM

3.1. The task of controlling a swarm of unmanned aerial vehicles

The task of controlling a swarm of unmanned aerial vehicles in this work is solved through a combination of centralized planning at the leader level and decentralized interaction between followers. The system is based on a leader-follower model, in which one of the agents (the leader) performs strategic navigation, forming a general route to the target, while the other agents (followers) adapt their behavior based on locally available information.

Leader trajectory planning

Leader trajectory planning is carried out using deep reinforcement learning methods, in particular Deep Q-Network (DQN), which allows for the complexity of the environment, including dynamic obstacles, to be taken into account. The leader, receiving the current state of the system (e.g., target coordinates, presence of threats, swarm configuration), determines the optimal actions taking into account long-term rewards, which allows adapting the route to changes in the environment and ensuring effective goal achievement.

Decentralized follower control

Followers in a swarm rely on information transmitted or received from neighbors about the leader's position or the swarm's direction of movement. Their decentralized behavior is based on simple heuristic rules or behavioral coordination methods, such as:

- maintaining distance between drones;
- speed alignment;
- avoiding collisions.

Each drone in the swarm, being a follower, is capable of acting autonomously within these rules, which allows maintaining a stable formation and adapting the swarm's movement to changes in the environment.

Obstacle avoidance at the level of each UAV

To ensure autonomous obstacle avoidance at the level of each UAV, a local algorithm based on artificial potential fields (APF) has been implemented. This method allows each drone to perform autonomous maneuvers in real time, avoiding obstacles without the need for constant data exchange with other drones or the leader.

Leader reassignment and fault tolerance

In case of leader loss, the system switches to role reassignment mode. SwarmManager or a built-in diagnostic module evaluates the parameters of candidates for the role of the new leader and initiates the transition with minimal disruption to the swarm structure. This mechanism ensures the stability of the swarm even in unstable conditions or when communication is lost, which is critical for complex missions where communication may be limited.

Integration of strategic planning and local algorithms

Thus, the task of controlling a swarm of UAVs is solved by integrating strategic planning at the leader level, local obstacle avoidance algorithms for each drone, and flexible swarm self-organization. This allows the swarm system to function effectively in conditions of limited information and a dynamic environment, ensuring high efficiency even in the presence of communication constraints and environmental obstacles.

3.2. Mathematical model of UAV swarm system

Let x_i (t) be the position of the i-th UAV, v_i (t) be its speed, and w be the corresponding weight coefficients, which are adjusted depending on the mission conditions. The total speed is formed as a weighted sum:

$$v_i(t) = w_{mig}v_{mig} + w_{obs}v_{obs} + w_{coh}v_{coh} + w_{sep}v_{sep}$$

where:

- Migration (interaction with the leader)

$$v_{mig} = x_L - x_i$$

- Obstacle avoidance (obstacle avoidance) – repulsive vector, reversed to the distance to the nearest obstacles

$$v_{obs} = \sum \frac{x_i - x_0}{\|x_i - x_0\|^2}$$

- Cohesion (tendency toward the center of the group)

$$v_{coh} = \frac{1}{|N_i|}\sum (x_j - x_i)$$

- Separation (avoidance of collisions with other agents)

$$v_{sep} = \sum \frac{x_i - x_j}{\|x_i - x_j\|^2}$$

The model provides both swarm behavior formation and dynamic response to changes in the environment, including obstacle avoidance and formation maintenance. Depending on the selected coefficients, it is possible to vary the degree of autonomy, the degree of swarm cohesion, and the nature of adaptation to external conditions.

3.3. UAV swarm control algorithm

The unmanned aerial vehicle swarm control algorithm implements a sequence of actions to ensure coordinated movement, mission execution, and fault tolerance. Interaction between agents is organized according to the leader-follower model and supplemented by local response mechanisms.

1. System initialization:

- A connection is established with all UAVs in the swarm.
- An initial leader is appointed based on system criteria (e.g., communication stability, charge level).

- The local coordinate system and the exchange of basic parameters (position, speed, identifiers) are initialized.

2. Leader route formation:

- The leader builds a global trajectory to the target using a DQN agent that optimizes the path taking into account obstacles, efficiency, and energy savings.

- The planning result is transmitted to followers in the form of control points or a direction vector.

3. Following the leader:

- Each follower determines its position relative to the leader or its closest neighbors and selects a direction of movement while maintaining the specified distance.

- A rule of alignment, clustering, and avoidance is used, implemented at the local level.

4. Local navigation and obstacle avoidance:

- Each UAV continuously processes sensor data (e.g., LaserScan) and generates an avoidance vector based on the artificial potential field (APF) method.
- If an obstacle is detected within the critical radius, the direction of movement is adjusted without disrupting the integrity of the formation.

5. Swarm and leader status diagnostics:

- SwarmManager (central or backup) periodically checks the communication, message relevance, coordinates, and operating mode of the leader.
- If failures are detected, a procedure for selecting a new leader based on specified criteria is initiated.

6. Mission completion:

- When the leader reaches the target area, the system initiates commands to complete the mission, such as landing, returning, or entering standby mode.

- All agents synchronize with the final state and enter the final mode.

3.4. Architecture of the software and hardware complex

The designed swarm control system for unmanned aerial vehicles is implemented as a decentralized architecture in which each agent functions as an autonomous unit capable of making decisions based on local information and limited inter-agent communication.

This approach allows for high adaptability, fault tolerance, and scalability of the swarm without dependence on a central control node. Physically, each UAV includes on board the main functional units necessary for independent performance of navigation, communication, and computing tasks.

The central element is a flight controller that stabilizes the aircraft, executes trajectory commands, and maintains the specified motion parameters. Navigation and sensor data processing, as well as the implementation of evasion, optimization, and inter-agent interaction algorithms, are performed on a separate computing module. It functions independently, without relying on external resources or a base station.

The hardware configuration is complemented by sensors that allow each drone to determine its own position in space and detect obstacles in real time. These include GPS receivers, inertial sensors, laser scanners, and cameras. Communication capabilities are implemented through wireless modules with a limited range, allowing each agent to maintain communication with its nearest neighbors within a given topology.

The software integrates navigation algorithms, a leader-follower model, obstacle avoidance mechanisms, and adaptive logic for switching between control modes. The leader generates a strategic trajectory based on a reinforcement learning algorithm, while followers perform tracking with local corrections and communication constraints. If the signal from the leader is lost, the system

automatically switches to a local mode of interaction with neighbors or reorganizes the formation with the appointment of a new leader.

The complex consists of the following functional modules:

- **Flight controller** (e.g., PX4, ArduPilot) — provides basic stabilization, position control, and trajectory command execution.
- **Computing module** (e.g., Raspberry Pi with ROS) — responsible for implementing DQN, APF, and WOA algorithms, local processing of navigation and sensor data, implementing local control algorithms, sensor data processing, inter-agent interaction, and providing an interface to the flight controller. ROS allows each functional component to be implemented as a separate node, which simplifies testing, updating, and scaling of the system.
- **Navigation module** — contains a GPS receiver, inertial system, and environmental sensors (e.g., LIDAR, cameras, ultrasonic sensors).
- **Interaction module** — performs the logic of information exchange between agents within a specified radius, ensures the formation of local clusters, supports topology, and transmits control messages. In this work, MAVLink is used as the basic protocol for implementing interaction between agents, which allows the transmission of status, position, and navigation target data in the form of lightweight binary messages.
- **Communication interface** — implemented on the basis of Wi-Fi or LoRa/NRF modules with the possibility of event-driven or periodic data transmission.

Thanks to this structure (see fig. 3.1), the system is capable of functioning in conditions of limited communication, high environmental dynamics, and unpredictable external influences. The decentralized architecture allows for

resilience to individual component failures and ensures a high level of autonomy for each agent in the swarm.

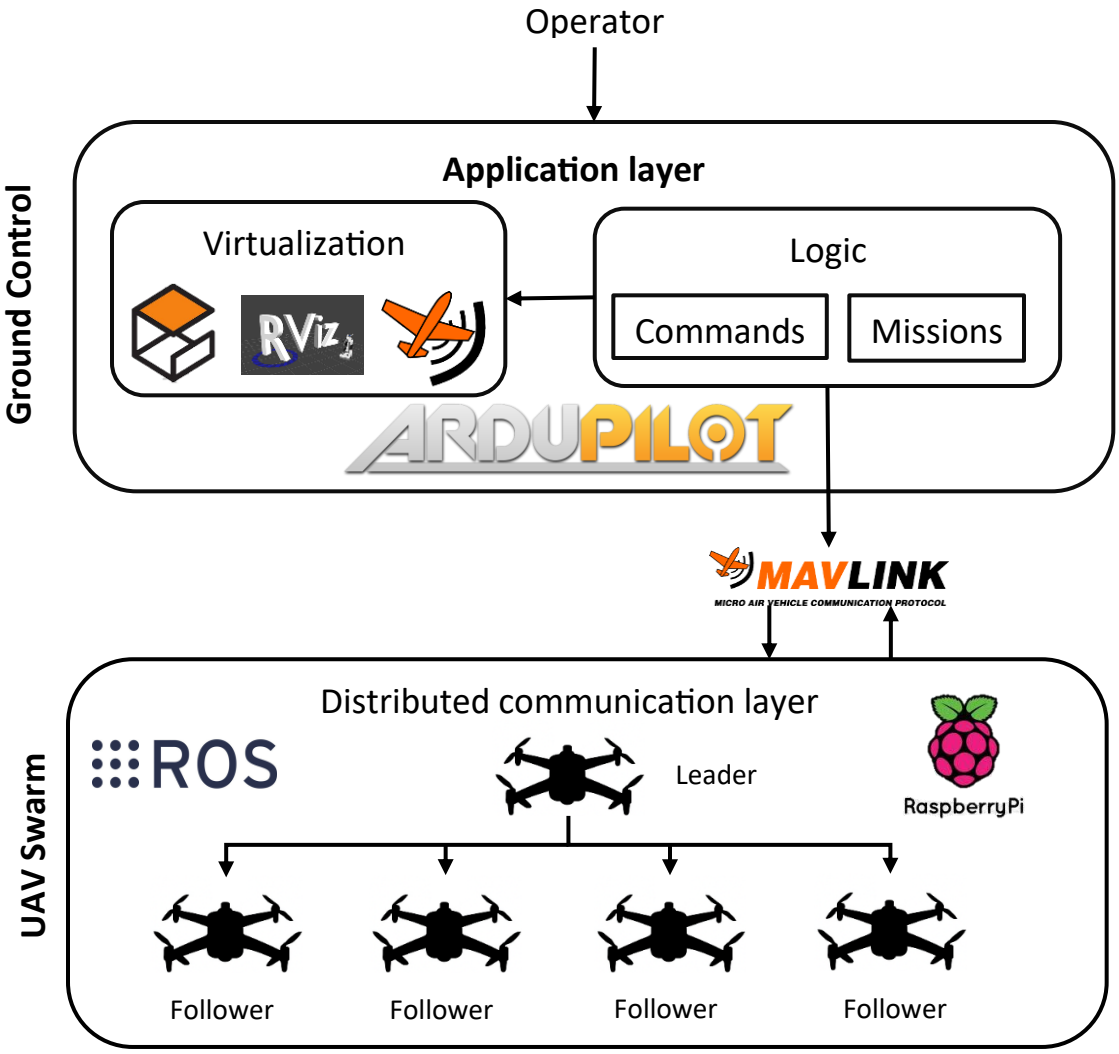

Fig. 3.1. Architectural design of a system with multiple UAVs

3.5. Application level

The application level in the proposed hardware and software architecture corresponds to the mission planning and high-level control of the UAV swarm. Its structure includes two main sublevels: virtualization (simulation) and logical (abstraction command).

Virtualization is implemented using open source software such as Gazebo, MAVProxy, and Rviz, which allow the behavior of drones to be simulated without the need for physical infrastructure. These include:

- MAVProxy is lightweight ground station software focused on controlling unmanned aerial vehicles via a command line. It allows you to launch missions, perform real-time telemetry, change flight view modes, and save logs.
- Gazebo is used for physically realistic flight simulation in a virtual environment with autopilot support. Thanks to its C++ API, it allows you to test complex algorithms without real hardware.
- Rviz allows you to visualize drone sensor data, both in simulation and in real flight.

The logical layer or abstraction layer provides high-level command processing, consensus algorithms, and coordination between agents.

It is based on the ROS framework, which provides convenient APIs for C++ (via roscpp) and Python (rospy). Thanks to this program, it can be deployed equally effectively on a virtual swarm during simulation and on real UAVs.

The connection between levels and agents is implemented through mavros, which is an interface between ROS and the MAVLink protocol. It allows scalable control of a swarm with many agents, ensuring the transmission of commands, services, and topics for interaction between the ROS nodes of each UAV. This creates conditions for a high degree of parallelism and distributed processing.

The component at this level focuses on solving complex tasks, including specific objects, mapping, and territory exploration. This allows the swarm system to be used not only for demonstration purposes, but also for real-world field applications.

3.6. Computational level

The computational level is responsible for distributing missions across the swarm, regardless of the leader-follower paradigm. The computational level of the swarm is decentralized and includes two types of UAVs, establishing a dyadic connection between leader UAVs and follower UAVs. It evolves over time in three phases: role assignment, role influence, and role routing. Role assignment is the initial phase in which the leader UAV assigns roles to follower UAVs and evaluates their status through feedback. Since leader UAVs are unfamiliar with followers in the initial phases of interaction, they can assess their availability based on their current states and the consensus error from the target state. Approaching the target state will increase the mutual attraction and trust between the leader and follower drones, leading to a reduction in consensus error. In the role assignment phase, leaders assign roles to followers if they consider these UAVs to be in good condition and meet mission expectations.

At the same time, follower UAVs will wait until they satisfy their current roles before accepting new role assignments. Finally, in the role routing phase, the communication between the leading UAVs and the follower UAVs is iteratively improved over time as they converge to a common state. The communication between the application layer and the swarm is carried out through the leading UAV. The swarm is fully connected and uses UDP-based protocols for communication between UAVs. It also receives a list of mission tasks and shares it with all leader UAVs according to the swarm topology (ring, star, or mesh).

In addition, the lead UAV serves as a gateway for transferring information between the GCS and the rest of the swarm. It is also responsible for swarm control and monitoring mission tasks across the swarm. When the operator sends a task, the lead UAV first requests a list of available processor cores on each lead UAV. It then sends the task as a mission to the available drones. For fault tolerance, a secondary leader UAV, also called the “backup leader,” provides fail-safe capabilities in case

the leader UAV fails. It regularly checks the status of the leader UAV and ensures mission continuity in case of problems. The UAV leader continuously sends a heartbeat signal to indicate its availability, followed by a set of status information. If, after a certain period of time, the backup UAV leader receives no information from the leader, it automatically switches from standby mode to active mode. This state occurs when the backup leader detects a negative event from the UAV leader. This may be due to an accident or hardware and software malfunctions.

On the other hand, follower UAVs simultaneously perform tasks assigned by the leader and periodically report on the status of the mission. It is important to note that the total time required to complete all mission tasks $M = \{j_1, j_2, \dots j_m\}$ depends on the number of drones k and processors p. When one UAV$_j$ completes its task $t_i^j \in j_i$, it waits for the other drones to complete their tasks. This principle allows the time complexity of mission M to be reduced by the number $(k-1)$ of UAV followers. Mission M, which must be performed with a swarm, is completely distributed and executed if the following is true:

- $\forall j \in [\![1..k]\!], i \in [\![1..m]\!], O(t_i^j) \neq 0$, and all ROS nodes are running.
- For a given UAV$_j$, nodes operating, t_i^j, t_l^j differ two by two, $t_i^j \cap t_l^j = \emptyset$.

For good parallelism, the correct number of ROS nodes n required to perform the task is calculated as follows:

$$n = \begin{cases} 0{,}95 \times k \times p, & \text{if homogeneous swarm} \\ 1{,}75 \times k \times p, & \text{if heterogenius swarm} \end{cases}$$

With a value of 0,95, we assume that the swarm is homogeneous, meaning that the UAVs have the same hardware components (rotors, grippers, batteries, etc.) and computing capabilities (RAM, chipset, and processor). At a value of 1,75, the swarm consists of different types of UAVs. The most efficient drones perform more tasks and launch a new wave of tasks immediately after their completion. Messages and services shared within the swarm are managed by ROS topics and services.

During computation, topics are published asynchronously and subscribed to by UAVs, allowing them to synchronize the called ROS services as a client or server.

3.7. Communication scheme

The communication scheme of the swarm is based on a single-group architecture. It is a decentralized scheme that facilitates communication between drones, thereby eliminating dependence on the GCS, as shown in Figure 3.2. The GCS communicates with the swarm via a drone leader, which acts as a “gateway UAV.” It controls and transmits data between the GCS and all UAVs using the MAVLINK protocol. Each follower UAV in the dedicated network is equipped with a point-to-point wireless communication device, allowing them to act as a single access point. Meanwhile, the leader's onboard communication module provides multipoint communication between the followers and the GCS. To support swarm scalability, we use a ring topology, which means that all bidirectional communication between two UAVs must occur through the leader drone. Since the communication infrastructure is supported by the MAVLink protocol, the swarm scales up to 255 UAVs or sensors.

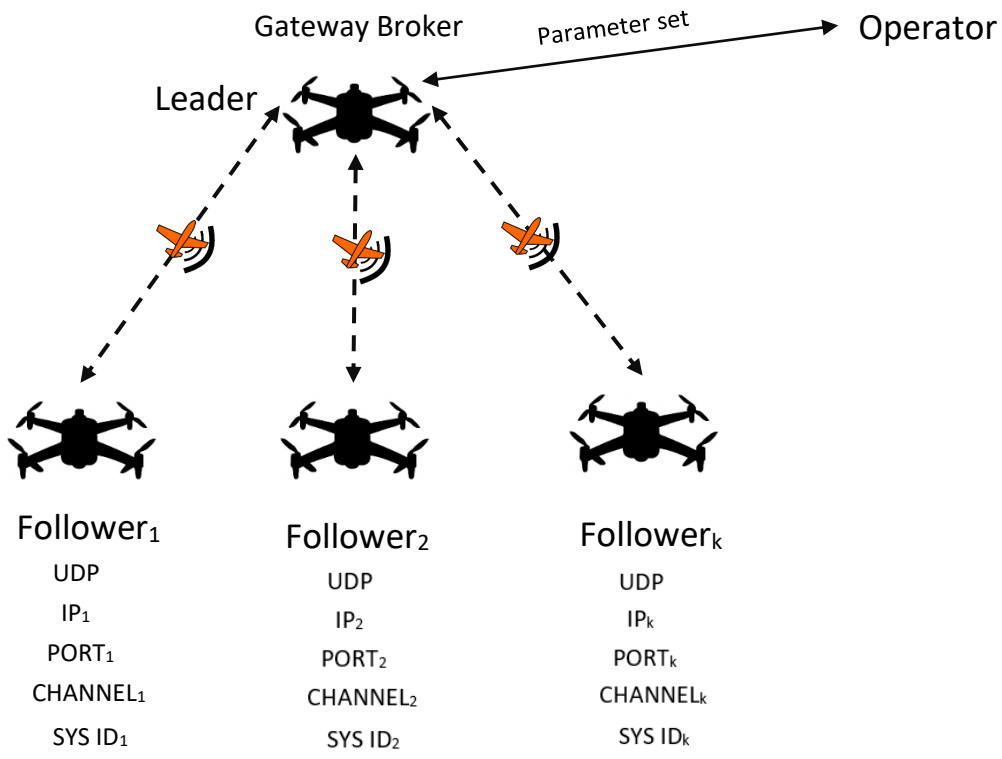

Fig. 3.2. Communication model of a swarm based on single-group architecture

The UAVs are integrated into the wireless swarm network via a bridge connection using UDP/Wi-Fi connections. Once the connection is established, a through path is created in the routing table. This routing table serves as the data transfer rules for each drone. The same process is repeated until k drones are integrated into the swarm. To ensure optimal message transmission and avoid packet collisions, each drone is identified in the wireless network by unique identifiers, which guarantee that each UAV is unique:

- SYS ID: System identifier assigned to each drone. Its value ranges from 1 to 255.
- IP: IP address used to identify a drone in the wireless network.
- PORT: Listening port used by the GCS and drones to send/receive packets and commands.

- CHANNEL: Communication channel used by the GCS to transmit telemetry data to the flight control unit of each drone.
- BAUDRATE: The speed at which each drone transmits and receives data.

As shown in Figure 3.3, communication within the swarm is based on message-oriented MAVLink protocols, where messages are sent with a built-in priority order according to different topics or channels. They occur at (1) the application layer, which defines the structure of messages exchanged within the swarm, and (2) the transport layer, which ensures reliable routing of messages. Each message is encoded in a packet structure that guarantees reliability at the transport layer. Message structures consist of a header, payload, and footer. The header contains identification and verification information, the payload stores the data itself, and the footer contains information about the integrity and security of the message. The semantics of the payload depend on the message types (control messages or data messages), which are highly context-dependent messages), which are highly

context-dependent:

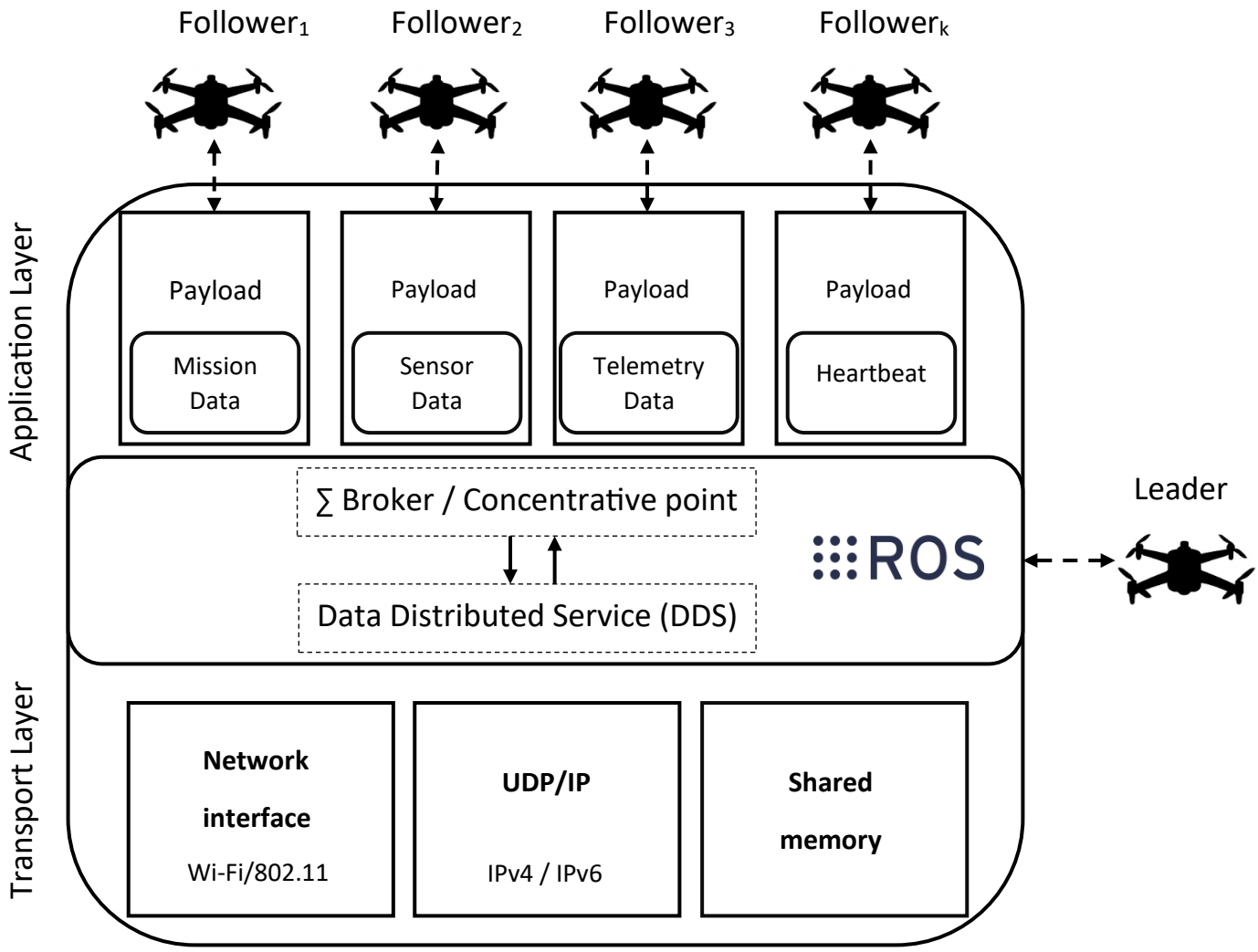

Fig. 3.3. Message interface for swarm communication via MAVLink

- Heartbeat signal – A periodic message that each drone sends at a lower frequency to indicate that it is active and functioning normally.
- Telemetry data provides feedback (waypoints, altitude, position, roll/pitch/yaw, speed, IMU, and diagnostics) about the drone's mobility.
- Flight control messages contain instructions (such as takeoff, landing, mode setting, RTL, etc.) that are used to send flight control commands with higher priority and frequency.

- Sensor information contains raw data from onboard sensors (e.g., GPS, GNSS) or cameras (e.g., thermal, spectral, or hyperspectral cameras).
- The parameter set contains standard definitions (e.g., HEARTBEAT, SYS STATUS, or PARAM VALUE) used to control a system with multiple UAVs.

Each UAV uses a separate channel to simultaneously broadcast these messages over the same communication channel. To distinguish itself in scenarios with multiple UAVs, each drone uses its own SYS ID and COMPONENT ID. All messages are concentrated and controlled by the lead UAV, which acts as a broker or concentration point, and data separation is achieved not by physical channels but by message identifiers and types.

3.8. Self-organization. Fault tolerance and fault bypass

Since the communication architecture of the swarm is based on a single-group architecture, it does not allow for fault tolerance. Therefore, high availability of the swarm depends on the SwarmManager. In case of failure or crash, the entire system with multiple UAVs will stop. In this context, we introduce a backup SwarmManager located on a separate leader UAV to eliminate the problem of single-point failure of the swarm. If the SwarmManager fails, the backup SwarmManager automatically takes over control.

Figure 3.4 shows the state transition diagram of the backup SwarmManager. It first issues diagnostic queries regarding the battery status, rotor status, and heartbeat frequency of the active leader UAV. If it detects any anomaly, it automatically switches to the active state and informs the sequential UAVs that it has taken control of the swarm.

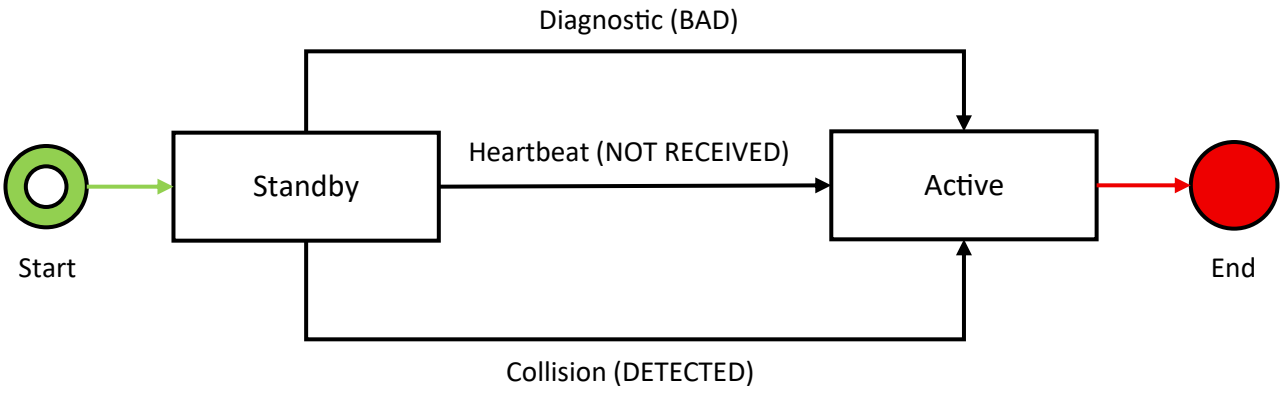

Fig. 3.4. State transition diagram of the backup leading UAV. The backup UAV enters active mode when one of three events occurs

During the mission cycle (see Figure 3.5), the active leader regularly sends signals to the backup leader, as well as a diagnosis of its health. At the same time, it regularly checks the status of the swarm and the missions being performed. This information is stored in a shared file accessible to the backup leader, who can use it to ensure continuity of operations in the event of an active leader failure.

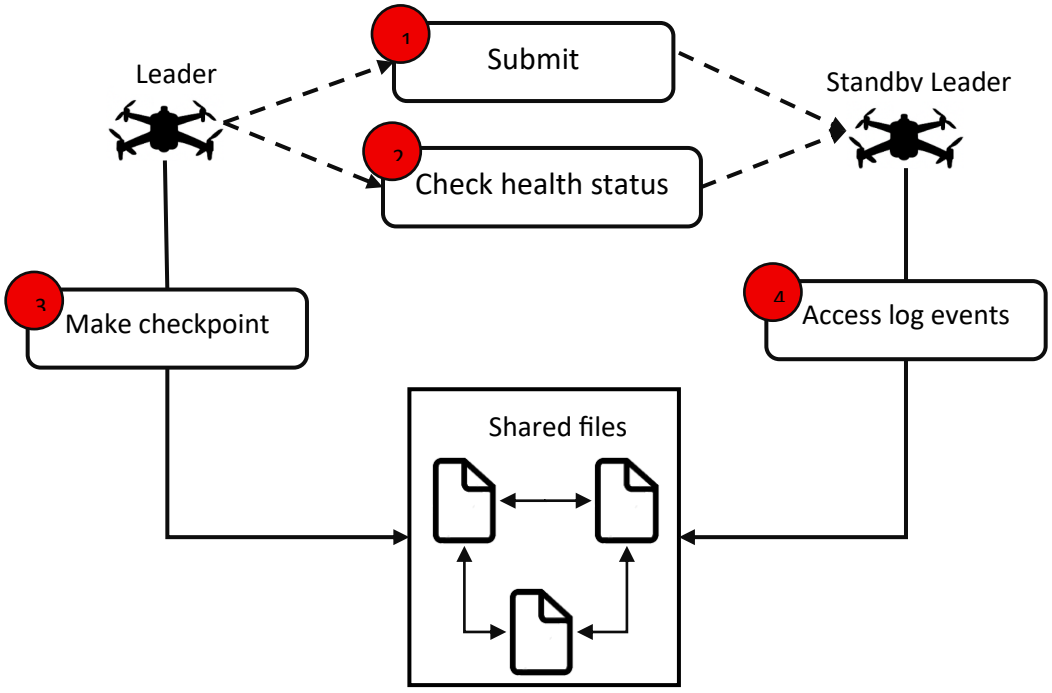

Fig. 3.5. Cooperative workflow of leading UAVs for fault tolerance policy management

During the execution of tasks by UAV followers, it is possible that in some situations the execution will fail due to computing power issues. In this case, the remaining tasks are reassigned to available UAV followers that have completed their tasks. This reassignment is performed in such a way as to take into account the swarm workload balancing constraints, so that each UAV follower has approximately the same workload w as the others.

CHAPTER 4
SYSTEM IMPLEMENTATION AND EXPERIMENTAL RESULTS

4.1. Software implementation of UAV swarm control

The software implementation of intelligent control of a swarm of unmanned aerial vehicles is based on the use of the Robot Operating System (ROS) environment, the ROS Noetic distribution, which is installed on each UAV computing module. The behavior of the devices is simulated in the Gazebo environment, which provides accurate modeling of the physics of motion and interaction of objects. To ensure communication with the autopilot, the MAVLink protocol is used in conjunction with the mavros package, which acts as a bridge between ROS and the PX4 flight controller.

Each drone in the swarm is implemented as a separate ROS node with unique parameters, including a system identifier (SYS ID), IP address, and communication port. The overall system architecture includes three key components:

- Leader – performs strategic planning, generates mission checkpoints, and coordinates the actions of followers.
- Followers – receive information from the leader and other agents and perform tasks taking into account local adaptation.
- SwarmManager (primary or backup) – monitors the overall status of the swarm, evaluates the leader's performance, and initiates task reassignment in case of failures.

Program code:

```
import rospy
from mavros_msgs.msg import State
from std_msgs.msg import String
```

```
# Callback function that is called when a message with the drone status is received
def leader_heartbeat_callback(msg):
    # Check: if the control mode is not OFFBOARD (standalone control)
    if msg.mode != "OFFBOARD":
        # Initiating the transition to a backup leader
        switch_to_backup_leader()

# Initializing the ROS leader node and subscribing to a topic /mavros/state
def init_leader_node():
    rospy.init_node("leader_node", anonymous=True)
    rospy.Subscriber("/mavros/state", State, leader_heartbeat_callback)
    rospy.spin()
```

This code is designed to monitor communication with the leader. If OFFBOARD mode, which means active autonomous control, is lost, the system switches to the backup leader. This is a basic example of a fault tolerance mechanism implemented at the ROS node level.

The leader_heartbeat_callback() function is designed to monitor the status of the leader drone. If it is detected that the active control mode is lost (i.e., msg.mode ≠ OFFBOARD), then the function to switch to the backup leader is activated.

The init_leader_node() function initializes a ROS node named leader_node and subscribes to the /mavros/state topic, which is the standard channel for exchanging information about the current flight controller mode.

Example launch configuration (launch file):

```
<launch>
  <!-- Initializing Gazebo -->
  <include file="$(find gazebo_ros)/launch/empty_world.launch">
```

```
    <arg name="world_name" value="$(find my_sim)/worlds/arena.world"/>
  </include>

  <!-- Leader Launch -->
  <node pkg="my_swarm_pkg" type="leader_node.py" name="leader1" output="screen">
    <param name="sys_id" value="1"/>
    <param name="ip" value="192.168.0.2"/>
    <param name="port" value="14550"/>
  </node>

  <!-- Launching two followers -->
  <node pkg="my_swarm_pkg" type="follower_node.py" name="follower1" output="screen">
    <param name="sys_id" value="2"/>
    <param name="ip" value="192.168.0.3"/>
    <param name="port" value="14551"/>
  </node>

  <node pkg="my_swarm_pkg" type="follower_node.py" name="follower2" output="screen">
    <param name="sys_id" value="3"/>
    <param name="ip" value="192.168.0.4"/>
    <param name="port" value="14552"/>
  </node>
</launch>
```

This launch file allows you to run a simulation with one leader and two followers, each with a unique SYS ID and network connection parameters, as required by the MAVLink protocol. This provides full inter-agent interaction within the virtual environment.

In addition to the main nodes, the system also includes:

- /mission_dispatcher — distributes mission targets among agents;
- /telemetry_monitor — monitors flight parameters;
- /backup_manager — activates a backup leader in case of failure detection.

This allows for high autonomy, fault tolerance, and scalability in UAV swarm control. All components interact via ROS, which provides flexible system configuration and the possibility of further expansion.

An obstacle avoidance script based on data from a laser scanner is integrated separately:

```
from sensor_msgs.msg import LaserScan
from geometry_msgs.msg import Point
from iq_gnc.py_gnc_functions import *
from math import cos, sin, pow, radians, sqrt

drone = gnc_api()

def laser_cb(msg):
    avoid_x, avoid_y, avoid = 0.0, 0.0, False
    for i in range(1, len(msg.ranges)):
        d0, k = 3.0, 0.5
        if 0.35 < msg.ranges[i] < d0:
            avoid = True
            x = cos(msg.angle_increment * i)
```

```
            y = sin(msg.angle_increment * i)
            u = (-0.5 * k * pow(((1/msg.ranges[i]) - (1/d0)), 2.0))
            avoid_x += (x * u)
            avoid_y += (y * u)

    if avoid:
        cr_heading = radians(drone.get_current_heading())
        avoid_x = (avoid_x * cos(cr_heading)) - (avoid_y * sin(cr_heading))
        avoid_y = (avoid_x * sin(cr_heading)) + (avoid_y * cos(cr_heading))
        dist = sqrt(pow(avoid_x, 2) + pow(avoid_y, 2))
        if dist > 3:
            avoid_x = (3 * (avoid_x / dist))
            avoid_y = (3 * (avoid_y / dist))
        cur_pose = drone.get_current_location()
        drone.set_destination(avoid_x + cur_pose.x, avoid_y + cur_pose.y, 2, 0)

def main():
    rospy.init_node("obs_avoider", anonymous=True)
    rospy.Subscriber("/spur/laser/scan", LaserScan, laser_cb)
    drone.wait4connect()
    drone.wait4start()
    drone.initialize_local_frame()
    drone.takeoff(2)
    rospy.spin()

if __name__ == '__main__':
    main()
```

The obstacle avoidance script implements reactive drone behavior based on data received from a laser scanner (LaserScan). After launch, the node initializes a connection with the autopilot and puts the drone into autonomous flight mode. During operation, it continuously receives information about surrounding objects, analyzing distances to obstacles. If objects closer than the set threshold are detected, a repulsion vector is calculated that simulates the effect of a potential field. This vector is transformed according to the current course of the drone, after which a new destination point is determined in a direction that avoids collision. The resulting point is transmitted to the autopilot as a new target for movement, ensuring effective obstacle avoidance in real time.

4.2. Simulation scenarios and experimental conditions

As part of the experimental verification of the performance of intelligent unmanned aerial vehicle control, a search and rescue scenario was implemented. In this model, a single UAV autonomously patrols a defined area using a snake-like coverage algorithm and stops the mission if a person is detected. Detection is performed using a YOLO-based neural network system (via the darknet_ros package), which reports the presence of a “person” class object. YOLO (You Only Look Once) is a convolutional neural network for detecting objects in images in real time. In the script, it is used indirectly through the Darknet ROS system, which processes the video stream from the drone's camera. The neural network analyzes the frames and outputs the coordinates of the detected objects.

The script is implemented based on the ROS framework, using iq_gnc utilities for flight control. The flight altitude is kept constant at 10 m. If a person is detected, the drone immediately lands, ending the mission early.

Initial conditions of the experiment:

- simulation environment: Gazebo;
- coverage area: 50 × 50 meters;

- flight altitude: 10 m;
- route type: snake-like trajectory;
- target detection method: /darknet_ros/bounding_boxes.

Code for an experiment:

```
#! /usr/bin/env python
import rospy
from darknet_ros_msgs.msg import BoundingBoxes
from iq_gnc.py_gnc_functions import *
from iq_gnc.PrintColours import *
from std_msgs.msg import Bool

mode_g = False  # False — search, True — rescue operation

def detection_cb(msg):
    global mode_g
    for bbox in msg.bounding_boxes:
        rospy.loginfo("{}% certain {} detected.".format(
            float(bbox.probability * 100), bbox.Class))
        if bbox.Class == "person":
            mode_g = True
            rospy.loginfo(CBLUE + "Person found. Starting Rescue Operation" +
CEND)
            rescue_pub.publish(True)

def follower_behavior():
    rate = rospy.Rate(2)
    while not rospy.is_shutdown():
```

```
            if rescue_signal:
                rospy.loginfo(CGREEN2 + "Follower entering rescue mode." +
CEND)
                # Add follower behavior in rescue mode
            rate.sleep()

        def rescue_signal_cb(msg):
            global rescue_signal
            rescue_signal = msg.data

        def leader_main():
            rospy.init_node("search_and_rescue_leader")
            rospy.Subscriber("/darknet_ros/bounding_boxes",          BoundingBoxes,
detection_cb)

            drone = gnc_api()
            drone.wait4connect()
            drone.wait4start()
            drone.initialize_local_frame()
            drone.takeoff(10)

            wps = []
            spacing = 10.0
            rows = 5
            drange = 50.0
            for i in range(rows):
                row = i * 2
```

```
            wps += [
                [row * spacing, 0, 10, 0],
                [row * spacing, drange, 10, 0],
                [(row + 1) * spacing, drange, 10, 0],
                [(row + 1) * spacing, 0, 10, 0]
            ]

        rate = rospy.Rate(2)
        i = 0
        while i < len(wps) and not mode_g:
            drone.set_destination(*wps[i])
            rate.sleep()
            if drone.check_waypoint_reached():
                i += 1

        drone.land()
        rospy.signal_shutdown("Search completed or person found.")

if __name__ == "__main__":
    try:
        role = rospy.get_param("~role", "leader")
        rescue_signal = False
        if role == "leader":
            rescue_pub = rospy.Publisher("/rescue_mode", Bool, queue_size=1)
            leader_main()
        else:
            rospy.init_node("search_and_rescue_follower")
```

```
            rospy.Subscriber("/rescue_mode", Bool, rescue_signal_cb)
            follower_behavior()
        except KeyboardInterrupt:
            rospy.signal_shutdown("KeyboardInterrupt")
```

The code implements a complete cycle of an autonomous mission. After initializing the ROS node, the drones ascend to a specified altitude and begin flying over the territory along a predetermined serpentine trajectory. The coordinates of the route points are generated dynamically using a for loop, where each pair of lines covers one horizontal strip of the search area.

The main detection mechanism is subscribing to BoundingBoxes messages from the /darknet_ros/bounding_boxes topic. The code receives coordinate data in the form of messages about object boundaries (BoundingBoxes). In the detection_cb callback function, each object is analyzed. If there is a “person” among the detected objects, a global flag mode_g = True is set, which interrupts the main movement cycle.

The drones then land. A message in the terminal informs whether the target was detected. If detected, the mission is terminated immediately. Otherwise, the drones complete their route.

Within this simulation, a search and rescue mission scenario was implemented in which drones patrol along a snake-like trajectory. Such a route is formed as a sequence of control points alternating between extreme coordinates with a fixed step. The visualization of this trajectory (Figure 4.1) demonstrates structured coverage of the specified area, which ensures effective scanning of the surface without gaps.

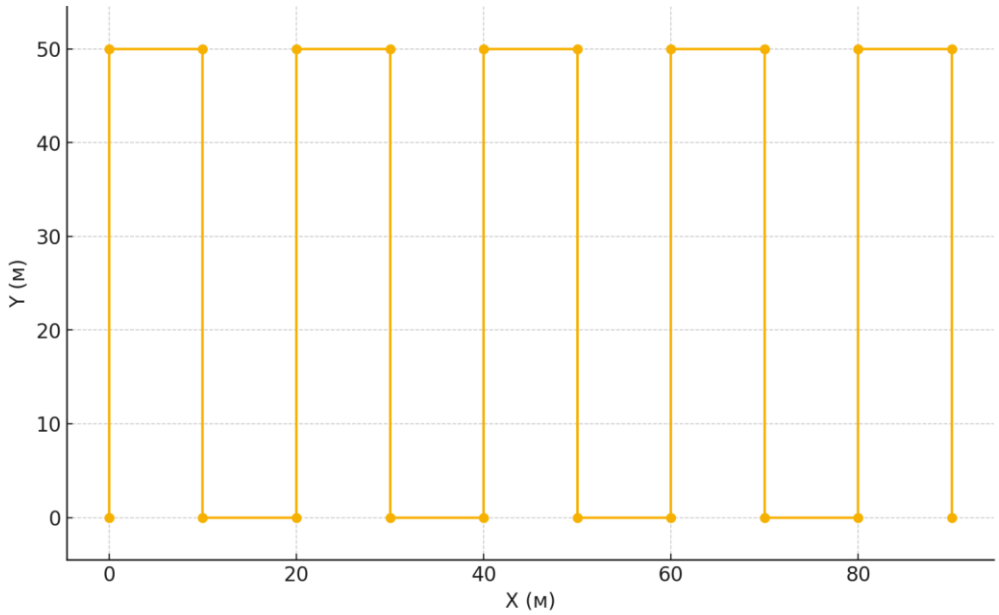

Fig. 4.1. Snake-like route of a UAV during search and rescue

The chosen snake-like movement strategy minimizes the number of turns while ensuring full coverage of the territory. This approach is often used in object detection missions, particularly in search tasks for people or infrastructure. The direction of movement changes only after reaching the extreme points of the specified range, which increases energy efficiency and reduces the load on planning computing resources.

It should also be noted that for small areas, adding more than two UAVs does not speed up flight time. This means that the tasks performed by UAV_3 and UAV_4 can be assigned to UAV_1 and UAV_2, respectively, with almost identical coverage times. This allowed us to conclude that the swarming concept is suitable for large reconnaissance areas.

4.3. Discussion of results

Modeling confirmed the effectiveness of the implemented intelligent UAV swarm control system under conditions of limited communication and interference. Reactive evasion algorithms integrated at the individual agent level demonstrated stable behavior in scenarios with dynamic objects, maintaining the specified

trajectory without collisions. Most importantly, the agents were able to function autonomously in case of loss of communication with the leader, proving the fault tolerance of the architecture.

The use of the Deep Q-Network algorithm in combination with preliminary route initialization using the Whale Optimization Algorithm ensured strategic planning at the leader level. The resulting trajectories were optimized for distance, collision avoidance, and energy consumption, demonstrating the system's ability to learn in a changing environment.

Another positive result was the detection and response to objects in the environment based on data from the YOLO neural network integrated via ROS Darknet. The search and rescue scenario demonstrated the system's ability to change the swarm's mission in real time depending on the event (human detection), confirming the flexibility of the implemented behavior logic.

CONCLUSION

Autonomous aerial swarms are of great interest to both industry and the scientific community because they offer a wide range of possibilities. Their creation is a challenge that involves several important tasks: communication, coordination, cooperation, and reconnaissance. The swarm must be fully equipped with a decentralized, scalable network infrastructure that allows new UAVs to be easily added and/or removed. Maintaining the quality of the communication signal is another challenge arising from the mobility of the swarm. As a result, UAVs must move in close formation to avoid signal loss, which is not uncommon due to collisions. We also need to define a set of rules based on strict consensus control that regulates collective decision-making, detects failures, and ensures system reliability.

Although various scientific works have already explored and investigated solutions to these problems, specific topics such as the mathematical formulation of swarm intelligence, self-organization, distributed mission planning, consensus control, and collective fault detection remain unexplored.

Experimental scenarios simulated in the Gazebo environment demonstrated stable swarm behavior when obstacles were detected and formation maintenance during autonomous movement. One of the key advantages is the implementation of the SwarmManager mechanism, which provides mission control and allows instant switching to a backup leader in case of communication loss or failure of the main leader. This has reduced the risk of complete loss of swarm control.

The LaserScan message-based obstacle avoidance script allowed for effective response to the environment and real-time route changes, while the search and rescue scenario with object (e.g., human) recognition demonstrated the ability of the swarm to automatically switch to response mode without direct operator control.

LIST OF USED REFERENCES

1. https://www.nature.com/articles/s41598-024-54531-w
2. https://www.mdpi.com/2504-446X/8/7/320
3. https://link.springer.com/article/10.1007/s42452-019-0551-z
4. https://www.sciencedirect.com/science/article/pii/S2666389922002367
5. https://www.mdpi.com/2504-446X/8/8/350
6. https://cdnsciencepub.com/doi/10.1139/juvs-2018-0009
7. https://www.sciencedirect.com/science/article/pii/S1000936121000534
8. https://www.mdpi.com/2072-4292/16/18/3490
9. https://onlinelibrary.wiley.com/doi/10.1155/2020/8530763
10. https://www.mdpi.com/2504-446X/9/2/134
11. https://ieeexplore.ieee.org/abstract/document/8323141
12. https://advancesincontinuousanddiscretemodels.springeropen.com/articles/10.1186/s13662-024-03841-4
13. https://www.mdpi.com/1424-8220/23/21/8766
14. https://www.sciencedirect.com/science/article/abs/pii/S1568494625000845
15. https://www.mdpi.com/2504-446X/8/10/575
16. https://www.mdpi.com/2504-446X/8/12/777
17. https://www.sciencedirect.com/science/article/pii/S2452414X18300086
18. https://www.sciencedirect.com/science/article/abs/pii/S2210650224001500
19. https://www.sciencedirect.com/science/article/pii/S1319157823004639

20. https://www.mdpi.com/2504-446X/6/12/402
21. https://www.mdpi.com/2504-446X/9/1/64
22. https://arxiv.org/abs/2412.12437
23. https://www.sciencedirect.com/science/article/pii/S1000936120301965
24. https://www.mdpi.com/2504-446X/8/6/226
25. https://www.mdpi.com/2504-446X/9/3/174
26. https://www.mdpi.com/2504-446X/7/9/267
27. https://www.sciencedirect.com/science/article/pii/S2212868920301118