UNIVERSITI
TEKNOLOGI
PETRONAS

SEPTEMBER 2024 SEMESTER

ALGORITHM AND DATA STRUCTURE

DRONE SWARM

LECTURER: M Nordin Bin Zakaria

No.	Name	ID	Program
1.	Muhammad Khairy Anwar bin Mohd Fariz	22006361	Computer Engineering
2.	Siti Nur Fatihah binti Imran	22007814	Computer Engineering
3.	Nurul Afiqah binti Rosli	22007301	Computer Engineering
4.	Mohd Izzudin bin Abdul Rahim	22003651	Computer Engineering
5.	Veebhaker Kalai Selvan	22007032	Computer Engineering

Table of Contents

Introduction

Drone swarming is a crucial part of autonomous systems and a primary focus in the study of algorithms and data structures (DSA). It entails the collaboration of several unmanned aerial vehicles (UAVs) or drones functioning as a cohesive unit. The orchestration of these drones presents distinct issues with real-time decision-making, communication, and data handling. This research examines the pivotal role of algorithms and data structures in addressing these difficulties, focusing specifically on their implementation in C# and the use of GitHub for collaborative development.

A drone swarm generally consists of several drones that interact and coordinate their actions to function autonomously. This enables them to execute many duties, such as aerial displays, surveillance, or other joint endeavours, with superior efficiency and efficacy. The dimensions and intricacy of these swarms may differ, although they are uniformly defined by their collective ability to carry out goals. Organizations such as Lumasky Drone Show, having substantial ability in the domain, offer significant insights into this sophisticated technology, proving the functionalities and prospective uses of drone swarming.

Technology Behind Drone Swarms

The technologies behind drone swarms form a wide range of hardware and software solutions that ease the coordinated operation of several types of UAVs. These systems have been implemented in many nations, and their use has markedly escalated in recent years. This section examines the technological elements of drone swarms, analysing how these components integrate to form a cohesive and synchronized aerial fleet:

1. **Communication Systems**: Drone swarms depend on resilient communication networks to send data and directives among individual drones. Communication can happen via multiple techniques, including radio waves, Wi-Fi, or alternative wireless protocols. The

instantaneous exchange of information is essential for the coordination of drone operations and manoeuvres.

2. **GPS and Navigation**: Each drone in the swarm employs GPS or other satellite navigation technologies to decide its exact location and height. The concept of satellite-based navigation has become essential for keeping aerial formation, allowing drones to perform tasks with exceptional precision.
3. **Sensors**: Drones are outfitted with various sensors, such as accelerometers, gyroscopes, magnetometers, and obstacle detection sensors (e.g., LiDAR or ultrasonic sensors). These sensors give information about the drone's orientation, velocity, and surrounding environment, easing secure and correct navigation.
4. **Onboard Computers**: Advanced onboard processors, in the innovative advancement of drone swarm technology, analyse data from various sensors and implement algorithms that govern the drone's flight trajectory and operations. These computers independently execute duties, ensuring formation maintenance, collision avoidance, and adherence to predetermined patterns or mission aims.
5. **Swarm Algorithms**: Advanced algorithms are fundamental to drone swarm technology. These advanced algorithms, supported by comprehensive studies, dictate the behaviour of the drones inside the swarm. Their abilities include sustaining designated formations, adjusting to evolving circumstances, and effectively carrying out mission goals. Swarm algorithms can be either centralized, governed by a single drone or ground station, or decentralized, wherein each drone interacts with proximate drones to reach collective judgments.
6. **Redundancy and Fail-Safe Mechanisms**: Drone swarms often integrate redundancy and fail-safe features to guarantee safety and reliability. Should a drone meet a technical malfunction, the swarm can reallocate jobs or adjust to support mission continuity.
7. **Real-Time Data Processing**: To guarantee safety and reliability, drone swarms often integrate redundancy and fail-safe systems. These steps, indicative of rigorous security and privacy protocols, are implemented if a technical malfunction in one drone. They ease the swarm's reallocation of jobs or adaptation to keep mission continuity effortlessly.
8. **Energy Management**: Effective energy management systems are crucial for prolonging the flight duration of drones inside the swarm. These systems may incorporate

sophisticated battery technology, power management algorithms, and the capability to recharge or exchange batteries during operations.

9. **Safety Protocols**: Safety is of utmost importance, and drone swarms are often equipped with safety rules to avert collisions, circumvent no-fly zones, and react to emergencies.
10. **Human-Machine Interface (HMI)**: Ground control stations or user interfaces enable human operators to oversee and regulate the swarm's behaviour. These interfaces ease situational awareness and allow operators to intervene as needed.

Algorithms in Drone Swarming

Drone swarming requires specialized algorithms to ensure efficient communication, decision-making, and coordination. These algorithms draw inspiration from natural systems and solve complex computational problems. Below are the key algorithms associated with drone swarming:

1. **Flocking Algorithms**: Inspired by the natural flocking behaviour of birds, drones follow rules that dictate their separation, alignment, and cohesion. This ensures drones keep a safe distance from each other while heading in the same direction toward a shared goal. These rules can be implemented in C# using object-oriented programming techniques, where each drone is modelled as an independent object.

2. **Optimization Algorithms (PSO)**: Particle Swarm Optimization (PSO) is another critical algorithm used in swarm intelligence. Drones use PSO to navigate complex environments and optimize their movement based on parameters like distance and fuel efficiency. Each drone, or “particle,” evaluates its performance in comparison to others and adjusts its trajectory based on this shared information.

3. **Pathfinding Algorithms**: To avoid obstacles and reach goals, drones use pathfinding algorithms like A* (A-star) or Dijkstra's algorithm. In a swarm, this process is more complex as each drone must consider not only its own path but also the paths of others in its proximity. The implementation of pathfinding can be done in C# by using custom classes that simulate grid environments or dynamic obstacle scenarios.

4. **Consensus Algorithms**: These algorithms ensure that all drones in the swarm agree on shared aims. Consensus algorithms often draw on distributed systems principles and can be represented using graph theory where nodes are drones, and edges stand for communication links.

Data Structures for Efficient Communication and Coordination

Implementing drone swarming also involves selecting proper data structures to ensure smooth and efficient operation. Some key data structures include:

1. **Graphs**: In swarming, graphs model the communication network between drones. Nodes in the graph represent drones, while edges are communication links. This structure allows for efficient traversal of the swarm and enables real-time decision-making. In C#, graph libraries such as QuickGraph can be used to model drone communication networks.

2. **Heaps and Queues**: Drones often need to prioritize tasks, such as mapping a target area or avoiding an obstacle. Priority queues and heaps help manage these priorities by efficiently storing and retrieving information. C# has built-in support for heaps (PriorityQueue) that can be used for these types of tasks in swarm simulations.

3. **Distributed Hash Tables (DHTs)**: DHTs are used to store and retrieve information in a decentralized manner, allowing drones to quickly access data shared by others in the swarm. This is critical in large-scale swarms where data must be distributed across multiple agents. Implementing DHTs in C# can be done using third-party libraries or by building custom implementations for specific drone swarming scenarios.

4. **Spatial Partitioning (Quadtrees)**: To avoid collisions and manage large-scale environments, drones often rely on spatial partitioning data structures like quadtrees or k-d trees. These allow drones to quickly query their surroundings and find nearby obstacles or fellow drones. In C#, spatial partitioning can be implemented to improve collision detection and reduce computational overhead.

Implementation in C# and GitHub

C# is an excellent language for implementing drone swarm simulations due to its object-oriented nature and support for complex data structures. Below are key aspects of how C# and GitHub are used in the project:

1. **Object-Oriented Programming**: Each drone in the swarm can be modelled as an object in C#. This allows for the clear abstraction of drone behaviour like movement, obstacle detection, and communication. Each drone object has properties such as position, velocity, and energy level, as well as methods for navigating and interacting with other drones.

2. **Real-time Communication**: C#'s event-driven architecture can simulate real-time communication between drones. Events are triggered when drones detect changes in the environment (e.g., an obstacle or a new target). This enables the swarm to adapt dynamically, reacting in real time to the environment.

3. **Collaborative Development using GitHub**: GitHub is used for version control and collaboration throughout the project. By creating a GitHub repository, the team can manage code changes efficiently. GitHub's branching and pull request system allows multiple team members to work on unique features (e.g., implementing a pathfinding algorithm or optimizing communication protocols) without interfering with each other's work. Continuous integration tools such as GitHub Actions ensure that code is tested and deployed smoothly after each change.

4. **Task Distribution**: Using GitHub, tasks can be assigned to individual team members with issues and pull requests, which helps organize contributions. The repository keeps a history of all changes, ensuring transparency and easing troubleshooting if bugs arise in the implementation.

5. **Simulation Testing**: GitHub also serves as a platform for managing simulation testing results. Drones are tested in virtual environments, and results from these tests (e.g.,

successful pathfinding, energy consumption) are logged and stored for future analysis. Each version of the simulation can be stored and compared to earlier iterations to evaluate improvements.

Simulating Drone Swarming in Unity with C#

For the simulation of drone swarming, **Unity** is an ideal platform due to its 3D environment capabilities and compatibility with **C#**. C# allows for efficient management of object-oriented programming, which is essential for simulating individual drones in the swarm. Each drone is programmed with specific behaviours like pathfinding, collision avoidance, and task management.

One crucial aspect of the simulation is performance measurement. As part of the project, we are taught to track the execution time of our C# programs using the Time class or Stopwatch function. Recording the time taken for specific tasks is essential for optimizing the code and ensuring real-time responsiveness, especially in large-scale simulations involving many drones. Efficient time management helps in adjusting algorithm performance, ensuring minimal latency during pathfinding, communication, and task execution.

Moreover, Unity's built-in physics engine allows for the simulation of real-world forces such as gravity and collision dynamics, giving a more realistic depiction of drone movement. However, managing the computational load of multiple drones is a challenge, where algorithms like *A** and Flocking for swarm behaviour must be carefully customized to reduce computational complexity.

Conclusion on Unity and C# Integration

The use of **Unity** and **C#** provides a powerful combination for simulating drone swarming behaviours in a 3D environment. Emphasis on **timing functions** is key to keeping performance efficiency, ensuring that the swarm runs smoothly without excessive computational delays. By using data structures and algorithms effectively, and monitoring performance, developers can optimize drone coordination and create more realistic simulations for practical applications.

References

Federal Government Accountability Office. (2023). *Science & Tech Spotlight: Drone Swarm Technologies.* Retrieved from https://www.gao.gov/products/gao-23-106930

Karatas, M., & Bilgin, H. (2022). Drone swarm systems and applications: A review. *Applied Sciences*, *14*(9), 3703. https://doi.org/10.3390/app14093703

Lumasky. (N/A). What is a drone swarm? *Lumasky.* https://lumasky.show/journal/what-is-a-drone-swarm/