ADVANCING DRONE SWARM SIMULATION: A COMPREHENSIVE SYSTEM FOR REAL-TIME CONTROL AND DYNAMIC INTERACTION

Dušan HERICH, Ján VAŠČÁK
Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic, E-mail: dusan.herich@tuke.sk

ABSTRACT

The rapid evolution of unmanned aerial vehicles (UAVs), has significantly advanced their capabilities, enabling complex operations that can be enhanced through swarm intelligence. This paper introduces a drone swarm simulator designed to model, analyze, and optimize the cooperative behaviors of drone swarms in diverse operational environments to provide a realistic and scalable platform for the simulation of drones, incorporating real-world physics, communication constraints, and autonomous decision-making algorithms.

Keywords: *drones, simulation, UAV, Internet of Vehicles*

1. INTRODUCTION

The rise of drone technology and its use in swarm robotics has brought advancements in research and operational capabilities. Swarm robotics, which draws inspiration from behaviors observed in nature such as bird flocks, fish schools, and ant colonies explores how multiple robots can work together in environments to achieve common objectives. The potential applications are extensive spanning from monitoring and disaster response to surveillance and environmental mapping [1]. Before deploying drones in real-world scenarios, simulating their behaviors within controlled environments offers benefits. It enables testing coordination algorithms, collision avoidance strategies, and task allocation methods among swarm members without the risk of equipment damage [2]. Furthermore, simulations can be conducted under conditions that may be challenging or impossible to replicate in reality providing insights into how swarms adapt to different challenges. However, developing swarm robotics systems comes with challenges. These include ensuring communication, among swarm members, thus designing algorithms for decentralized decision-making processes and creating physically robust robots capable of operating across diverse environments. Additionally, the dynamic nature of swarms—where collective behavior emerges from robot interactions—requires modeling and simulation capabilities.

Considering these difficulties the main goal of our simulation system is to offer a platform, for exploring experimenting, and improving swarm robotics ideas. Specifically, our system aims to:

- Model realistic drone physics and behaviors within a virtual environment.
- Allow for the simulation of complex interactions between drones.
- Offer tools for the real-time control and monitoring of drone swarms.
- Facilitate the testing of different control algorithms and communication strategies under a variety of conditions.

When looking at existing systems there is a range of focuses and technologies to consider. For example, SwarmLab, created by [3] is a software tool based on MATLAB that simplifies algorithm testing, fine-tuning, troubleshooting, and performance evaluation, for drone swarms. It allows for comparing algorithms for drone swarms in environments and offers metrics to measure swarm behaviors. To set apart our simulation system, from existing platforms like SwarmLab it's crucial to highlight the elements of the proposed approach. While SwarmLab as outlined by [3] provides simulation features within MATLAB our system goes further by integrating Blender for 3D modeling and Python for intricate control logic. This combination enriches the intricacies of the drone swarms and also enables a deeper level of real-time interaction and responsive environmental dynamics. The selection of these technologies aims to craft a simulation environment that closely mirrors real-world complexities providing an interactive tool, for studying drone swarm behaviors.

Other systems [4] also focus on designing a simulation system for the aerial vehicle (UAV) swarms that delves into analyzing various network technologies crucial to UAV swarm operations like data gathering and communication networking. Their framework highlights the simulation features of UAV swarm systems with an emphasis, on technical performance aspects of UAV swarm networks.

This paper has been structured to describe the process of developing a simulation system starting from the concept, to implementation and evaluation. After this introduction, Section 2 goes into detail about the methodology used, explaining how different technologies were integrated and discussing the foundations of the simulation. In Sections 3 and 4, we provide an overview of the system design, including modeling drones and environments and incorporating engines to ensure realistic simulations. Section 4 focuses on discussing the control mechanisms that were developed to interact with drones within the simulation environment. Moving on to Section 5 we explore the software tools and modeling techniques that were employed in building and optimizing the simulation environment. Finally, in Section 6 we evaluate the results of our simulation efforts offering insights into both system performance and its potential applications, in swarm robotics research.

2. METHODOLOGY

The creation of a simulation system, for swarm robotics, involves integrating algorithms, physical models, and real-time interaction capabilities. In this section, we will discuss the foundations, technological integration, and practical considerations that form the basis of our simulation systems design and functionality.

At the core of the proposed simulation is the concept of swarm intelligence, a phenomenon where collective behaviors emerge from rules followed by individuals. Inspired by biology our system can model interactions based on principles like self-organization decentralized control and adaptability. These principles guide the development of algorithms that govern drone behaviors ensuring that our simulated swarm exhibits dynamics [5]. When discussing the proposed system coordinates and controls drones it's important to differentiate it from systems like [5] which uses particle swarm optimization (PSO) for pathfinding in UAV control highlighting the effectiveness of PSO in managing UAV navigation and task assignment aligning with our goal of simulating drone interactions and behaviors. However, our simulation incorporates those concepts into an environment created in Blender with support from Python, for control and flexibility. This setup allows to test PSO-based algorithms in dynamically changing scenarios as well as to explore their scalability and efficiency in more complex swarm situations.

The simulation system utilizes frameworks to model drone coordination and control. This can include employing pathfinding algorithms like A* for navigation purposes utilizing optimization techniques such as particle swarm optimization for task allocation and implementing consensus algorithms to ensure swarm behavior [6].

To construct an interactive simulation environment we have utilized Blender as our primary platform. Blender's versatility in modeling, texturing, and animating serves as a foundation, for creating both the drones themselves and the laboratory setting in which the swarm operates. The environment in which the drones operate is not static. It consists of elements such as moving obstacles and multiple drones, which adds complexity and challenges for the swarm to navigate [7].

To control the drones in time during simulations we have developed a customized interface using Python. This interface communicates with the Blender environment through Python sockets allowing us to send control commands and receive feedback instantly. Our focus while designing this system was to ensure delay and high reliability mimicking the experience of controlling a swarm of drones [8].

Scalability has been a factor in developing our simulation system. It can accommodate numbers of drones ranging from groups to large swarms consisting of hundreds of units. This scalability also extends to simulating task complexities and environments making our system useful for research purposes and contexts. Our strategies, for optimization crucial for maintaining high-performance simulation capabilities are based on model simplification. The system designed in [11] delves into drone model classification using networks (CNN) trained on synthetic data. It highlights the importance of accurately simulating a variety of drone models. This not only impacts model identification but also plays a role in enhancing realism and performance.

Maintaining reproducibility in experiments conducted within the simulation is essential for rigor. Our system includes mechanisms for logging and replaying scenarios enabling researchers to replicate conditions and outcomes. This capability is vital, for validating swarm robotics experiment results.

By taking into account factors like scalability and reproducibility the system provides a platform to delve into the possibilities of swarm robotics.

3. SYSTEM DESIGN

Developing a simulation system that accurately captures the dynamics of drone swarm behavior and their interaction, with environments involved careful planning and execution, hence, in this section we present an overview of the design process, which includes creating models of drones and the environment and addressing challenges related to simulating real-world physics.

Ensuring the fidelity of drone models is pivotal to achieving useful simulation results. Drone models were meticulously crafted to replicate the characteristics and flight dynamics observed in real-world drones. We paid attention to factors like size, weight, and propulsion mechanisms when designing components such as rotors, bodies, and sensors using Blender. This enabled us to mimic their interactions within the simulation environment mirroring those found in the realm [9].

The simulation environment extends beyond a backdrop; it is an interactive space that both influences and is influenced by the drones. To evaluate drone navigation algorithms and collision avoidance strategies effectively we modeled a laboratory equipped with obstacles like walls and moving objects. Additionally, we simulated factors such as varying lighting scenarios to analyze their impact on swarm behavior.

To ensure real-world-like behaviors within our simulation framework we integrated Blender Game Engine physics engine. This allowed us to apply principles, like gravity, momentum, and aerodynamic drag to our drone models accurately. The engine also helped us simulate collisions, between drones and obstacles in the environment [10].

In this simulation, we used models to replicate the flight dynamics of drones. This included factors like lift, thrust, drag, torque, and even the stabilization systems of the drones. We implemented control algorithms to manage these dynamics effectively allowing for control over drones as well as the entire swarm.

Developing this simulation system required finding a balance between realism and computational limitations. To overcome this obstacle we utilized optimization techniques such as polygon reduction for models and level of detail rendering (LOD). These strategies ensured that the simulation could run smoothly in time without compromising its quality too much.

Simulating interactions within a drone swarm posed challenges especially when it came to obstacle navigation and coordination between drones. To address this we developed algorithms inspired by natural swarm behaviors. These algorithms followed rules of separation, alignment, and cohesion which allowed for swarm dynamics to emerge from interactions among individual drones.

Through the creation of drone models and a thorough understanding of their surroundings, coupled with the integration of a reliable physics engine and the successful resolution of considerable design obstacles we have created a platform that provides insights, into the dynamics of drone swarms [12]. This system serves as a valuable testing ground for real-world applications, across diverse fields.

The communication module (Figure. 1) of the simulation system comprises a server and several clients. In order for a client to connect somewhere, a server must be running. In this system, the server is started at the same time as the simulation is started. After it is started a socket is created with predefined parameters (addressing type and socket type). The socket then needs to be bound to a specific address and port. In the next step, the server listens for incoming connections from clients. Next, the server must accept the incoming connection. In this step, the server is looped until a defined number of clients connect. Once a client is connected, the server returns a socket to the client that represents their connection and allows communication.

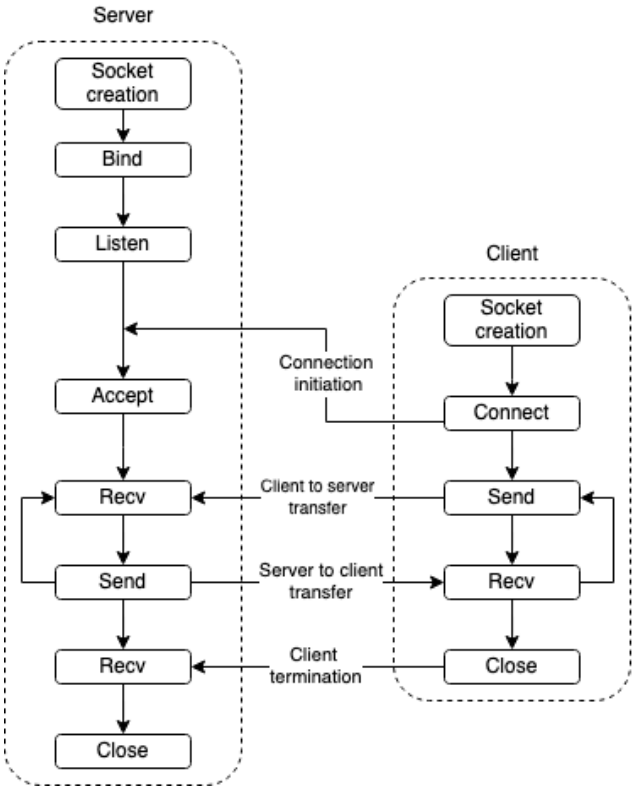

Fig. 1 Client-server communication

Once the connection is successfully established, both parties can communicate with each other. When controlling a drone, the client always sends a message first and the server then replies with a confirmation message with the drone's current position in space. The sending and receiving of messages is done in an infinite cycle until the termination condition is met.

The whole part of the client (Figure 2) is largely run in a simple cycle. In the actual beginning, the socket initialization takes place, when the client connects to the server in order to send control commands via the keyboard. The cycle begins waiting for input from the user. Once the keyboard is pressed, validation takes place and a key from the list of allowed keys is detected.

In the server part (Figure 3), it starts with initialization, which consists of finding the drone and camera objects to work on next and creating two threads running alongside the main thread. The main thread takes care of running Blender, and thus for other work it should not be used for any other work - if it were used, it would most likely halt the program and terminate it. The first thread created takes care of receiving messages from the client. The second one in turn handles the processing and execution of the received commands. The thread for receiving messages initializes the socket at the very beginning and waits for the client to connect. After a successful connection, it listens for incoming messages on the designated port. When a message is received, it validates it and queues it and sends it to the other thread. The given thread, which is also running a loop, reads the message and performs the necessary action based on the key. This can be e.g. moving forward, turning left, changing the camera, ending the simulation, etc. After a successful command from the user is executed, it is returned to the client The current position of the drone is sent back to the client.

The design of the control interface prioritized the needs of the end user aiming to create a responsive experience that reflects the complexities of operating real-world drones (Figure 4 and Figure 5). Key considerations included making it easy to use and accessible while also providing users with information, about drone status and environmental conditions. The layout of the interface was designed to offer commands and real-time visualizations of data allowing operators to make decisions quickly.

Operators can control the drones through a command panel that supports a range of inputs from movements to complex coordination strategies for swarm operations. The panel allows for both the control of drones and the execution of pre-programmed behaviors for swarm operations. Advanced features include the ability to adjust simulation parameters on the go such as conditions and obstacle configurations which facilitates scenario testing.

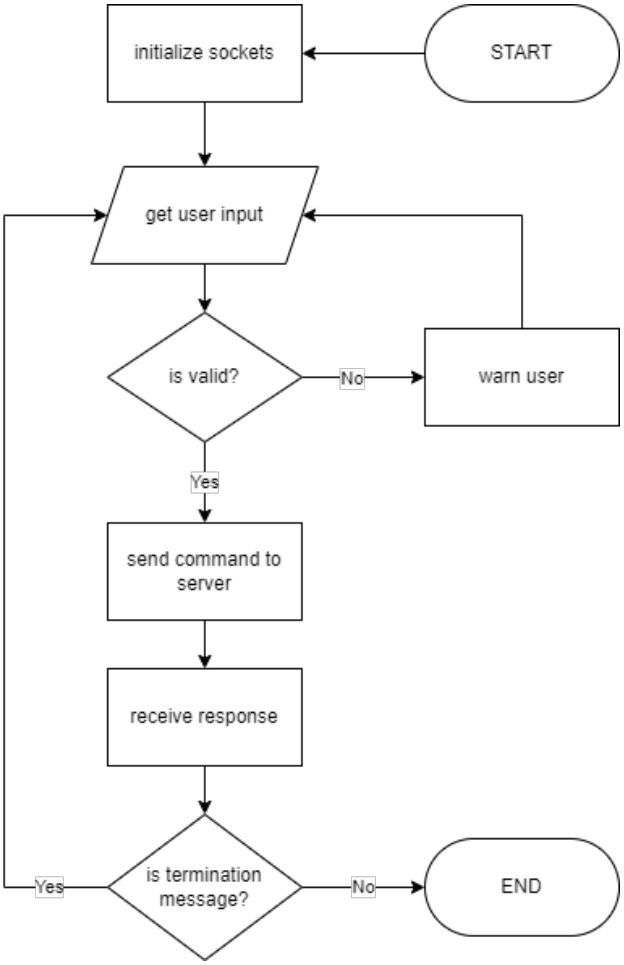

Fig. 2 Client process during the communication

The control mechanism relies on a communication protocol established between the control interface and the simulation environment. By utilizing Python sockets, this system ensures data exchange enabling real-time transmission of commands to the drones and providing feedback to operators. This bidirectional communication is essential for responsive control over the swarm [13].

Overcoming challenges was necessary to achieve real-time performance in the simulation. This involved optimizing data transmission to minimize delays and developing algorithms of processing complex control commands. Through testing and refinement, we engineered the system to maintain performance even when dealing with large swarms and dynamic environmental conditions [13].

The control mechanisms were designed to be adaptable and scalable accommodating simulation scenarios and swarm sizes. We achieved this flexibility through an architecture that allows for the integration of new control algorithms and expansion of the interface's capabilities. Scalability considerations ensured that the system could handle increased demands from swarms without compromising responsiveness or user experience [13].

Our simulation systems control mechanisms represent an advancement in simulating drone swarms by providing users with an interface, for precise control and feedback.

Furthermore, to develop a lifelike simulation, for drone swarm research we need several software tools and modeling techniques.

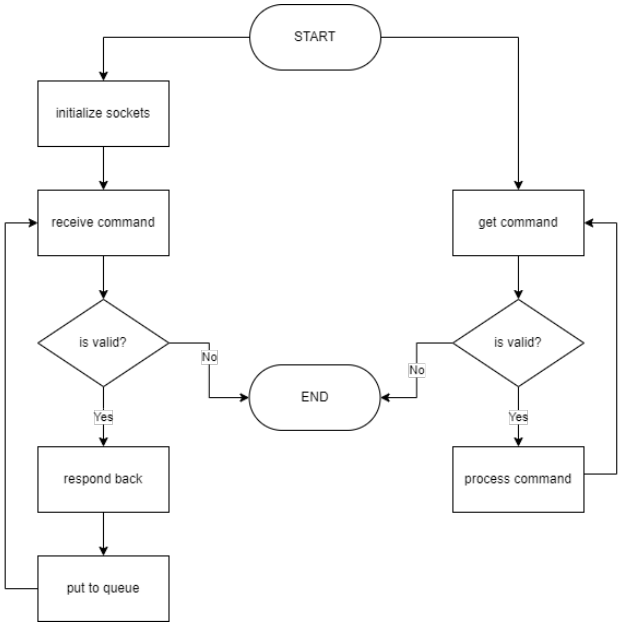

Fig. 3 Server process during the communication

Blender is a software tool that we rely on for its range of features, in 3D modeling texturing, and animation. We chose Blender because it offers a set of tools and its source. With Blender, we can create models and environments and take advantage of its powerful animation capabilities. It was the choice for building both the drones and the dynamic simulation environment. The integrated physics engine in Blender also allowed us to apply movements and interactions within our simulated world.

When it comes to scripting Python played a role due to its versatility and seamless integration with Blender. We used Python scripts to automate aspects of the simulation from controlling drone behavior algorithms to managing dynamics. Additionally, Python socket programming capabilities enabled real-time control between the operator and the simulation.

To ensure behavior that closely resembles real-world physics we put great emphasis on crafting detailed drone models. We employed techniques like mesh editing and sculpting to model drone geometries. Applying materials and textures added realism, to their appearance enhancing the quality of our simulation.

Our simulation environment was carefully designed to replicate a laboratory setting with obstacles and variable conditions. We utilized modeling techniques to develop objects and dynamic elements such, as moving obstacles.

Considering the high resource requirements for simulating models and environments we employed optimization strategies as an aspect. We simplified models by reducing polygon counts wherever without compromising quality significantly. Additionally, we implemented Level of Detail (LOD) rendering to adjust model complexity based on their proximity, to the camera. This improved performance while maintaining a user experience.

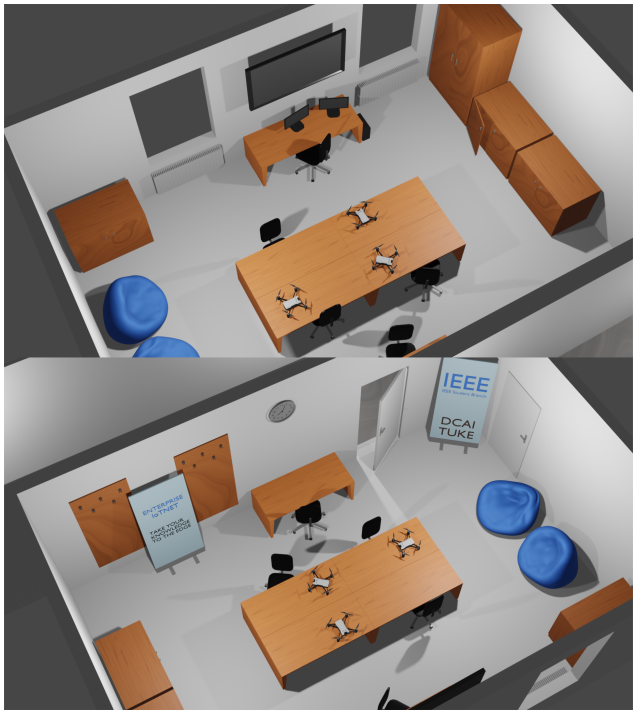

Fig. 4 Model of the laboratory

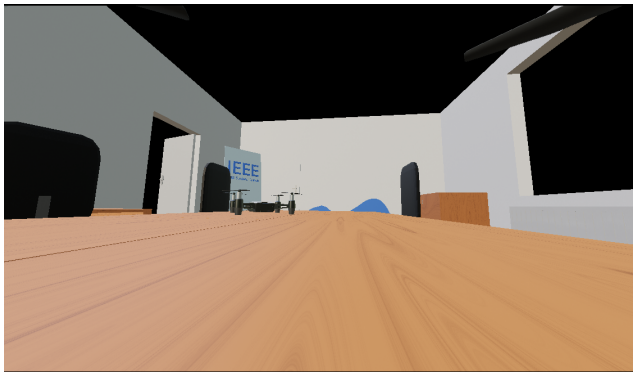

Fig. 5 POV mode from the drone' camera

It was crucial to optimize the Python scripts to maintain real-time performance especially when dealing with swarm interactions. We improved the scripts to minimize burden and streamlined data handling processes for communication, between the simulation and control interface.

4. EVALUATION

A crucial step, in the advancement of the proposed simulation system involved evaluating its outcomes which was vital to not just validate the system's capabilities but to pinpoint areas that needed improvement. In this section, we will discuss the methods employed to assess the simulation's performance evaluate the authenticity of drone behaviors, and determine the system's usefulness as a research tool.

The performance of the system was assessed based on metrics, like frame rate and responsiveness to real-time control inputs. Our primary focus was to ensure the simulation operates at an optimal frame rate, targeting 60 frames per second (FPS) to facilitate a seamless experience. Achieving an average FPS range of 50-60 in complex simulations with multiple objects underscores the system's robust performance capabilities (Table 3).

The scalability tests, pivotal in demonstrating the system's capability to handle increased load without compromising performance, revealed significant outcomes. By employing model simplification - rendering with various rates of decimation, and Level of Detail (LOD) techniques, we observed a balanced reduction in resource consumption while maintaining simulation integrity. For instance, using the collapse method on a drone model resulted in a decrease in polygons (single points in 3D space), vertices (two-dimensional shapes making up the surfaces of 3D models), and tris (triangles, as the simplest form of polygons), efficiently lowering computational demands without severely impacting visual fidelity (Table 2, Table 3).

Table 1 Comparison of achieved FPS

Experiment	No. objects	No. polygons	FPS
joined objects	44	97 607	50-60
parent-child	187	97 607	< 5
dec. modifier	44	30 566	80-100
materials	44	30 566	85-100

Table 2 Applying the Collapse method to a drone

Rate	No. polygons	No. vertices	No. tris
0.4	25 695	16 512	33 099
0.3	19 133	12 333	24 825
0.2	12 796	8 158	16 549
0.1	6 921	3962	8 274

Table 3 Applying the Un-Subdivide method to a drone

Rate	No. polygons	No. vertices	No. tris
1	21 908	21 293	42 736
2	15 468	12 907	25 962
4	11 315	8 072	16 323
5	10 813	7 436	15 053

5. FUTURE ENHANCEMENTS AND DIRECTIONS

The evaluation of the proposed simulation system provided valuable insights, for further development. These findings pave the way for improvements that can greatly enhance the usefulness and applicability of the system.

Simulating environments with accuracy presents both challenges and opportunities for swarm robotics research. Our future work will focus on modeling cityscapes allowing us to study drone swarms in applications such as delivery services, traffic monitoring, and infrastructure inspection.

A significant direction for research lies in integrating advanced AI-based control algorithms into our system. By leveraging machine learning techniques like reinforcement learning we can develop drones capable of learning and adapting to complex environments and tasks. This will greatly enhance the autonomy and efficiency of our swarm.

The simulation environment provides a platform, for training AI models allowing for controlled experimentation with various learning algorithms. Our future goals include developing an interface to integrate AI models making it easier to train and test drone behaviors within the simulation.

As interest grows in drone swarms we recognize the need to simulate swarms with hundreds or even thousands of drones. We will dedicate efforts towards optimizing our simulation's performance so that we can study swarms without compromising real-time responsiveness or detail.

In conclusion, simulation holds a role in advancing swarm robotics research. To continue serving as a tool for for further research, we need to focus on enhancing environmental complexity integrating AI and machine learning capabilities improving realism and performance, and ensuring accessibility. As we delve into improving our simulation system with AI and machine learning features the importance of addressing latency and real-time performance grows significantly. The study [14], on optimizing latency in UAV-enabled MEC systems for virtual reality applications aligns well with our objectives. Managing latency, as highlighted in their research is crucial for providing a simulation experience especially as we incorporate more sophisticated AI-driven control algorithms. By learning from these advancements our goal is to reduce communication and processing delays, in our system to ensure that the simulation remains responsive and accurate even as we tackle environmental scenarios and enhance drone autonomy.

In summary, future research should be focused on:

- Modeling complex environments like cityscapes to study drone swarms in urban applications such as delivery services, traffic monitoring, and infrastructure inspection.
- Integration of advanced AI-based control algorithms.
- Optimizing the simulation's performance to support larger swarms, enabling the study of swarms of drones without compromising detail or real-time responsiveness.
- Improving the simulation's realism, environmental complexity, and user accessibility.

These future directions have the potential to enhance drone swarm capabilities.

ACKNOWLEDGEMENT

This publication was supported by the VEGA grant EDEN: EDge-Enabled intelligeNt systems (1/0480/22).

REFERENCES

[1] CHEN, W. – LIU, J. – GUO, H. – KATO, N.: Toward robust and intelligent drone swarm: Challenges and future directions. *IEEE Network*. **34**, 278-283 (2020)

[2] CASADO, R. – BERMÚDEZ, A.: A simulation framework for developing autonomous drone navigation systems. *Electronics*. **10**, 7 (2020)

[3] SORIA, E. – SCHIANO, F. – FLOREANO, D.: SwarmLab: A MATLAB drone swarm simulator. *2020 IEEE/RSJ International Conference On Intelligent Robots And Systems (IROS)*. pp. 8005-8011 (2020)

[4] LI, W. – HU, Y. – WANG, F. – WANG, C. – ZHANG, W.: Simulation System Design of Unmanned Aerial Vehicle Swarm. *2022 IEEE International Conference On Unmanned Systems (ICUS)*. pp. 109-114 (2022)

[5] PYKE, L. – STARK, C.: Dynamic pathfinding for a swarm intelligence based UAV control model using particle swarm optimisation. *Frontiers In Applied Mathematics And Statistics*. **7** pp. 744955 (2021)

[6] QAMAR, S. – KHAN, S.– ARSHAD, M. – QAMAR, M. – KHAN, A.: Autonomous drone swarm navigation and multi-target tracking in 3D environments with dynamic obstacles. *ArXiv Preprint ArXiv:2202.06253*. (2022)

[7] LEE, S. – YANG, J. – LEE, B.: Drone simulation technologies. *Electronics And Telecommunications Trends*. **35**, 81-90 (2020)

[8] SAFFRE, F. – HILDMANN, H. – KARVONEN, H.: The design challenges of drone swarm control. *International Conference On Human-computer Interaction*. pp. 408-426 (2021)

[9] GRIGOROPOULOS, N. – LALIS, S.: Simulation and digital twin support for managed drone applications. *2020 IEEE/ACM 24th International Symposium On Distributed Simulation And Real Time Applications (DS-RT)*. pp. 1-8 (2020)

[10] ROCCA, R.: Fault animation with 3D model integrating drone and satellite images. *EGU General Assembly Conference, 2021. Proceedings. Doi*. **10** (2021)

[11] WISNIEWSKI, M. – RANA, Z.: & Petrunin, I. Drone model classification using convolutional neural network trained on synthetic data. *Journal Of Imaging*. **8**, 218 (2022)

[12] HASSIJA, V. – CHAMOLA, V. – AGRAWAL, A. – GOYAL, A. – LUONG, N. – NIYATO, D. – YU, F. – GUIZANI, M.: Fast, reliable, and secure drone communication: A comprehensive survey. *IEEE Communications Surveys & Tutorials*. **23**, 2802-2832 (2021)

[13] HASSAN, G. – HUSSIEN, N. – MOHIALDEN, Y.: Python TCP/IP libraries: A Review. *International Journal Papier Advance And Scientific Review*. **4**, 10-15 (2023)

[14] NASIR, A.: Latency optimization of UAV-enabled MEC system for virtual reality applications under rician fading channels. *IEEE Wireless Communications Letters*. **10**, 1633-1637 (2021)

Received February 16, 2024, accepted May 14, 2024

BIOGRAPHIES

Dušan Herich graduated (MSc) in 2021 at the department of Cybernetics and Artificial Intelligence of the Faculty of Electrical Engineering and Informatics at Technical University of Košice and is currently a PhD student with research focused on intelligent transportation.

Ján Vaščák is an Associate Professor in the Department of Cybernetics and Artificial Intelligence at the Technical University of Košice, Slovakia. He earned his MSc. in 1990 and his Ph.D. in 1996 from the same university. His research interests are primarily centered on intelligent control and mobile robot navigation, including computational intelligence, adaptive fuzzy systems, neural networks, evolutionary computing, decision support systems, multi-agent systems, cooperative robotics, the Internet of Things, and ubiquitous robotics.