DronePort: Smart Drone Battery Management System

Zdeněk Bouček[0000−0002−0152−1188], Petr Neduchal[0000−0001−5788−604X], and Miroslav Flídr[0000−0002−5338−2351]

Faculty of Applied Sciences, New Technologies for the Information Society
University of West Bohemia, Univerzitní 8, 306 14 Plzeň, Czech Republic
{zboucek, neduchal, flidr}@kky.zcu.cz
http://fav.zcu.cz/en/, http://ntis.zcu.cz/en/

Abstract. This paper deals with the description of a drone management system for long-term missions called DronePort. First, the issue of long-term missions and possible approaches are outlined. Further, the individual components of proposed system, both hardware, and software are introduced. The DronePort system relies on battery swapping. By storing the battery in a battery compartment, the system is not strictly designed for one type of drone, but with simple modification, it is capable of maintaining a flight of various Vertical Take-Off and Landing (VTOL) drones. Afterward, more attention is paid to the simulation environment, which will greatly facilitate the development of the entire system. The simulation includes both drones equipped with a down-facing camera and a DronePort landing platform, which is fitted with an ArUco marker for precise landing. Next, the DronePort Traffic Control system is presented, which is tasked with communicating with the drones, scheduling battery swapping, and planning trajectories for the flight to and from the DronePort landing platform. The system uses the standard MAVLink protocol for communication, enabling use with a variety of MAVLink compatible drones. Finally, an example of collision-free trajectory planning considering battery capacity is presented. Trajectory was found in terms of Chebyshev pseudospectral optimal control.

Keywords: Aerial robotics · Drones · Battery management · Traffic control · Robot simulation

1 Introduction

Nowadays, one of the biggest challenges in the drone field is the long-term missions with multiple drones, which brings new demands for autonomy [1, 2]. The vast majority of drones use Lithium-Polymer (LiPo) batteries, and usually, they last for only several tens of minutes in the air. In most applications, a hard-wired drone is out of the question because of the limited operating area and the increase in weight of the entire system due to cables.

There are applications with wireless charging such as [2], nevertheless, this solution is more time demanding than the standard battery swapping method

and requires staying on the ground. The standard method is to allocate the person who will operate the charging station. However, the employment of drones is usually considered to minimize the involvement of human operators to increase workplace safety and minimize personnel costs.

The state-of-the-art research shows that the most efficient method, which solves the power supply and charging issues, is autonomous battery swapping. Several experimental solutions are tightly adapted to specific drone models and batteries [3, 4]. DronePort (DP) system proposed in this paper can operate various drones with Vertical Take-Off and Landing (VTOL) ability. It is capable of swapping batteries of various sizes and shapes.

The article is structured as follows. The subsequent section outlines the entire DP system with a description of its parts. Next, the simulation environment is described to streamline the development of the whole system. Then the DronePort Traffic Control system is introduced, and an example of trajectory planning is presented.

2 DronePort System

The DP system is an early-stage project with the goal of automating long-term drone missions. The development of the system involves the University of West Bohemia and SmartMotion, where SmartMotion is mainly responsible for the creation of the entire DP landing platform and battery storage. The system consists of two main parts. The first part is hardware, which consists of a landing platform and a battery compartment placed on the drone.

The landing platform is equipped with an ArUco code that allows safe and precise on-spot landing. It also includes battery storage and a smart multi-slot battery-charging station. The size and weight of the DP landing platform allow it to be moved by two people. A robotic manipulator is used to remove the battery from the drone, plug it into the charging station, and plug the charged battery into the drone. The battery compartment allows easy handling by a robotic manipulator and can be supplemented in the future with an NFC tag for easy identification of the battery by the system.

The DP system will also be equipped with a computer capable of connecting with the drones, downloading the necessary information, scheduling, and sending commands with battery swapping missions. In order to achieve wide usability of the system, it will use the standard MAVLink protocol [5] for communication. We intend to use drones equipped with Pixhawk PX4 autopilot for testing, but it is possible to interact with all drones communicating via MAVLink. So far, the PX4 software-in-the-loop (SITL) simulated drone in Gazebo has been successfully withdrawn from the mission to the desired coordinates, and the mission has been backed up in case of problems. Afterward, the simulated drone took off and successfully resumed its mission. The drone communication application was implemented in Python and was based on the pymavlink[1] library.

[1] https://github.com/ArduPilot/pymavlink

Next, the software part of the project will be briefly described. Drone state estimation is not performed by the DP system. This fact makes the system very versatile and does not require a specific set of sensors. The only requirement is a down-facing camera due to the detection of the ArUco code on the landing platform. The system will estimate the battery's state, which will also be affected by the long-term behavior of the battery. This state will also be predicted based on drone and smart charger data for optimal scheduling of battery replacement.

Another software component is the landing controller, where the orientation and position data of the ArUco code is used. The next component is the DronePort Traffic Control system (DPTC), which, based on information received from the drones and the DP landing platform, decides on the scheduling of battery swapping and plans trajectories for an approach to the DP landing platform and subsequent return to the mission. DPTC will be discussed in more detail in Section 4. Another essential component is the simulation of the whole system, which serves as a tool for testing all parts of the DP system in a virtual environment and will be described in more detail in the following section.

3 Simulation

In order to reduce the cost and effort of the task, the simulation environment is used. In this section, we describe and discuss the design of the DronePort simulation. A brief overview of available simulators capable of simulating one or multiple drones will be presented in the first part. As will be mentioned, the Gazebo simulator, together with the PX4 controller, was chosen for our experiments. In the next part, DronePort simulation in Gazebo simulator[6] will be described. The third part of this section focuses on the unique DronePort model generation developed to reduce modeling effort.

3.1 Available open-source simulation software

Based on our research and several survey papers on this topic, such as [7], [8], [9], [10], we put together the following list of simulators capable of simulating missions of one or multiple drones. The well-known simulator is Gazebo which is based on Ogre 3D graphic engine. It can be easily extended using plugins written in the C++ programming language. Moreover, it can be connected to the PX4 controller using PX4-SITL_gazebo[2] plugin suite.

Several simulators are built on the top of the Gazebo simulator, such as RotorS[11] or BebopS[12]. It is a set of models, sensors, and controllers that can communicate with the Gazebo simulator.

There also exist simulators that are not built on the top of the Gazebo. The following projects can be mentioned. The first one is open source AirSim [13] simulator developed by Microsoft. Its interesting feature is the photorealistic graphics which enables to solve machine vision problems inside the simulation.

[2] https://github.com/PX4/PX4-SITL_gazebo

Another interesting fact is that the AirSim can be run using Unreal Engine or Unity as its 3D graphics engine. Similar photorealistic graphics based on the Unity engine is available inside FlightMare [14] simulator developed by Robotics and Perception Group at ETH Zurich. Another option with photorealistic graphics is NVIDIA ISAAC[15] which is primarily intended to use NVIDIA Hardware.

Finally, more general simulators such as V-Rep[16] or Webots[17] can be mentioned. We chose the combination of the Gazebo simulator and PX4 controller for our experiments because it offers a well-known extensible environment with sufficiently simplified graphics to perform traffic control experiments with a DronePort model.

3.2 DronePort simulation

DronePort model for simulation is simplified. The model's geometry is created using SDF[3] specification used by Gazebo simulator and consists of a blue block of model body and of the thin block textured using ArUco code. The model is shown in 1.

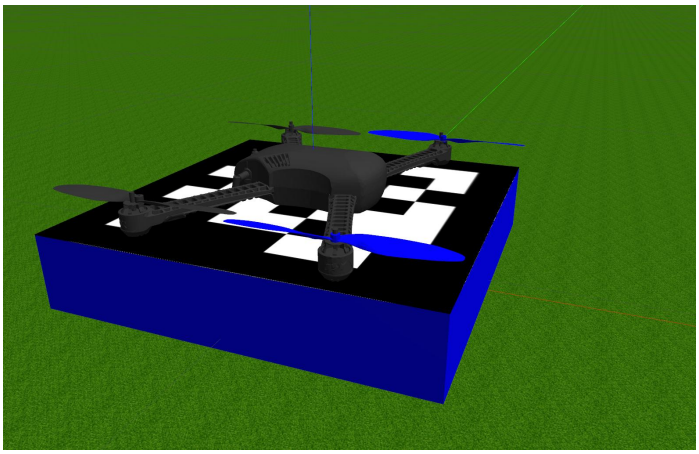


Fig. 1: Example of DronePort object and Iris drone in Gazebo simulator.

The software for the DronePort model is shown in Fig. 2. It consists of several parts that will be described in the following text. The main part is a DronePort Control Software that can be implemented in an arbitrary programming language. It handles the state of the DronePort device – either simulated or real. The control software is connected to the rest of the system and with a simulator using MAVLink protocol – see MAVLink Router block in the scheme. As visible from the scheme, MAVLink Router is connected with both the PX4 controller

[3] http://sdformat.org/spec

for communication with drones and Gazebo Plugins to communicate with the simulated world in Gazebo. The last part of the scheme is a camera plugin connected to a simulated drone. It provides source image data for the ArUco code detector, which is used during the landing maneuver of the drone.

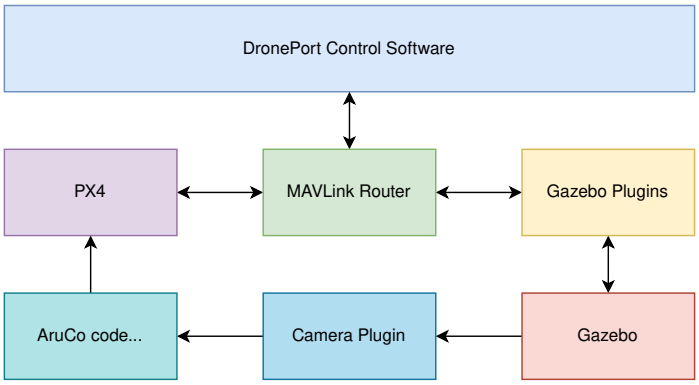


Fig. 2: DronePort simulation scheme

The DronePort model in the Gazebo is equipped with a Gazebo plugin that broadcast its GPS coordinates using MAVLink message. This plugin is based on the gazebo_mavlink_interface plugin from the mentioned PX4-SITL_gazebo plugin suite.

3.3 Generation of a unique DronePort model

During the development of the DronePort model, the first intention was to create a unique DronePort during the initialization of the simulation. Unfortunately, it was discovered that it is no simple task to set texture with ArUco code on the fly. Thus, the generator of unique models from the template was developed in Python. Its purpose is to generate a model with particular properties such as ArUco code, size, color, and MAVLink protocol settings. Moreover, it can be connected to the simulation using software like Tmux and Tmuxinator, and thus, it can be launched before the simulation starts.

In practice, the simulation is based on the clear Gazebo without Robot Operating System (ROS). On the other hand, the support for ROS can be easily added to the model using a particular plugin. With a model generator, it is possible to run either simple or complex scenarios with one or multiple drones and one or multiple DronePort devices. Thus, a simulation of traffic control with DronePort can be performed. More about traffic control will be described in the following parts of the paper.

4 DronePort Traffic Control

This section describes the DronePort Traffic Control (DPTC) system, which is complemented by an example of collision-free drone trajectory planning concerning battery capacity is presented. The main task of the DPTC is to schedule the replacement and subsequent recharging of drone batteries at the DP platform. It will do so by monitoring and predicting the state of drone batteries and registering the number and state of batteries available on the DP platforms. Opposed to [3], the DPTC will optimally schedule the interruption and resumption of the current drone missions, considering the DP landing platforms availability, availability of charged batteries at the DP platforms, and minimizing mission interruption time. The DPTC will take care of the trajectory re-planning to the DP platform according to the current mission progress and the resumption of the original mission after battery replacement.

The functionality of the entire system can be inferred from the data flow for the DPTC API shown in Fig. 3. From DPTC's perspective, the whole system is divided into four parts. The first part is Ground Control (GC), which is not manipulated by DPTC and is only used to operate the drone. The second part is the drone itself which contains a battery compartment. The drone communicates with the GC and sends information to the DP platform. DPTC will be able to operate several drones at the same time. The drone contains low-level control and mission control, which executes commands from the GC and DPTC. The third part is the battery compartment, which contains an NFC tag to identify the battery. For each battery, the system makes a prediction of the state based on the measurement. The last part is the DronePort platform, which includes smart battery chargers that perform charging and measure battery status. The computer in the DP platform runs the DPTC, which schedules drone withdrawals from missions and handles communication with the drones. The activity of the DPTC from the drone's perspective is illustrated in Fig. 4.

The design of the data structure used to store the information used in the DPTC is shown in Fig. 5. The data should contain information about all the essential parts of the DP, i.e., drones, landing platforms, and charging stations. About the batteries, the current parameters and their history could be stored. In the case of drones, their battery ID, position, status, or mission plan could be stored. Data about a DP platform could include its location, geofence, and a list of components (batteries, chargers, assigned drones). For the time being, the system is considered fully centralized due to the fact that only one DP landing platform is considered. However, in the future, more platforms could be involved in the system, even with several DPTCs, where drones would be serviced, for example, based on airspace zoning and the occupancy of individual platforms.

Further, an example of collision-free drone trajectory planning will be demonstrated. The trajectory will be planned considering the State of Charge (SoC) of the battery [18]. The trajectory planning problem will be described as an optimal control problem (OCP) [19]. First, the Chebyshev pseudospectral method (PSM) will be introduced and employed to acquire the solution to the OCP. Sub-

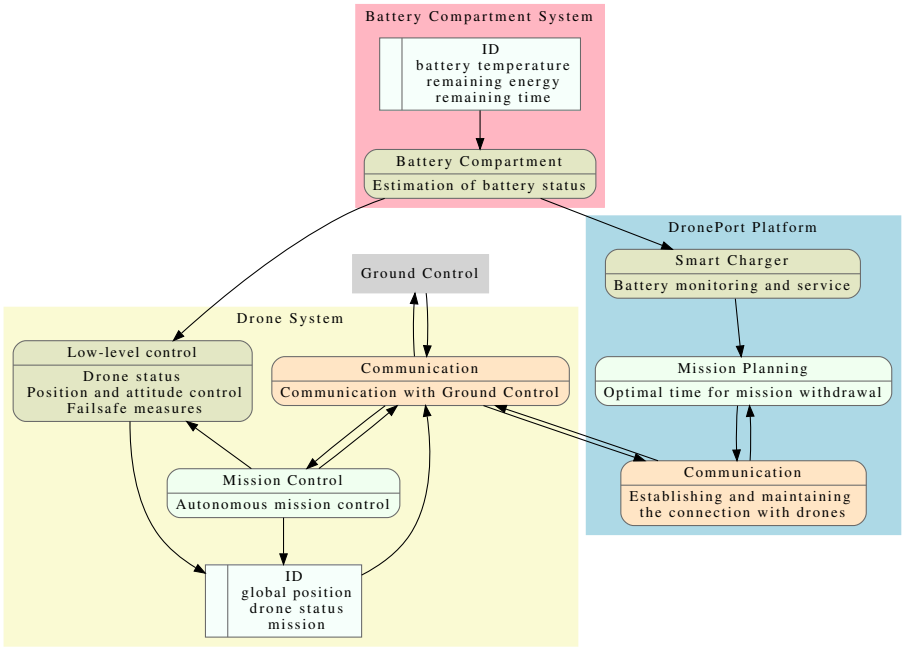


Fig. 3: Data flow of DronePort Traffic Control system API

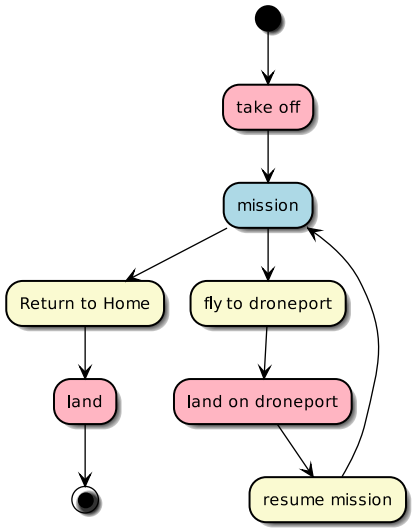


Fig. 4: Drone behavior

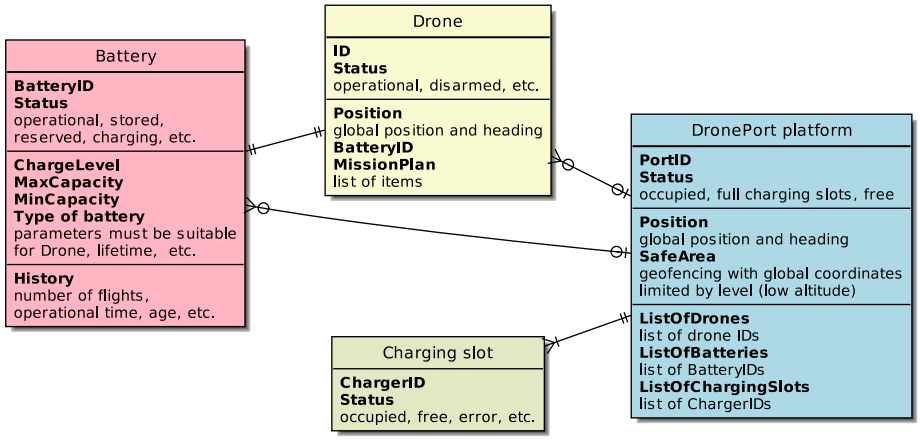


Fig. 5: Proposed Data Structure

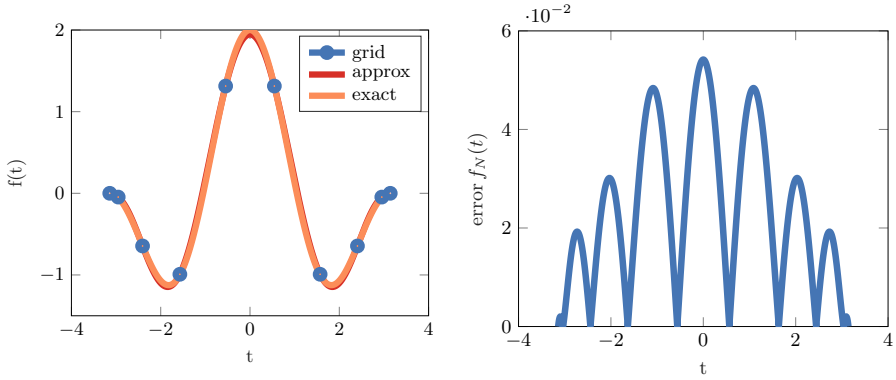
sequently, the parameters of the problem will be described. Finally, the solution will be presented.

4.1 Chebyshev Pseudospectral Optimal Control

In this part, the Chebyshev pseudospectral method (PSM) [20–22] will be presented in the context of its utilization in solving the OCP. The PSM gives an exact solution to the problem at "grid" or so-called "collocation" points. It approximates the function outside these points by a basis set of functions usually composed of trigonometric functions or polynomials. Approximation of function and its error is clearly shown in Fig. 6. Each basis set has its unique set of optimal collocation points. The points are usually the roots of function, which can be augmented by boundary points. The presence of boundary points is essential for boundary value problems such as OCP.

The standard Chebyshev polynomial of the first kind [21] with a grid containing the boundary points is chosen. The integral objective is approximated with Clenshaw–Curtis quadrature, and the derivative which is needed for approximating dynamics is calculated using differential matrix. The Chebyshev approximation can be used on the interval $[-1, 1]$. Therefore, it is necessary to transform the problem coordinates according to the end-points of the time vector when solving the OCP.

PSM generally achieves higher accuracy at lower complexity than Finite Element Method or Finite Difference Method and better solution convergence than shooting methods [21]. Therefore, it is advantageous to use it for drone trajectories as it allows fast computation when re-planning is required and low complexity to be used even on a drone up-board computer.



(a) Function

(b) Error of approximation

Fig. 6: Approximation of function with Chebyshev pseudospectral method

4.2 Trajectory Planning

This part will describe the parameters of the trajectory planning problem and also provide implementation details. The state of the system consists of the position and speed of the drone and the State of Charge (SoC). The control vector is composed of the acceleration in the x, y, and z axis. The dynamics of the drone are described by a simple model. The x-axis motion is described as

$$\dot{r}_x = v_x,\ \dot{v}_x = a_x, \tag{1}$$

where r_x, v_x, and a_x are position, velocity, and acceleration in x-axis, respectively. Movement in the other axes is described in the same way. State of battery is described in a standard manner using SoC which is given as 0.0 and 1.0 for empty and full battery, respectively. The dynamics of SoC is described as

$$\dot{b} = B_b \left(v_x^2 + v_y^2 + v_z^2\right) + D_b, \tag{2}$$

where b is the battery SoC, the linear discharge versus time is represented by the parameter D_b and B_b is the coefficient of dependence on the square of the drone velocity. Each state is bounded by a box constraint. The problem contains fixed boundary points except for SoC b and end time t_f, which have a free end-points. Objective function includes SoC maximization.

The problem is further supplemented with constraints for obstacles in the form

$$-\left|r - c_i\right|_2 + R^2 \leq 0, \tag{3}$$

where r is position of drone, c_i is center of i-th obstacle and R is a radius of obstacle. Obstacles are sampled randomly through the whole space with uniform distribution.

4.3 Results

The trajectory planning problem was implemented in Python using the Pyomo optimization toolbox [23]. The apparatus for the Chebyshev polynomial approximation was rewritten based on the MATLAB implementation in [20]. The solution was sought on a standard desktop PC with Intel Core i9-9900 and Ubuntu 20.04 using well-known open-source NLP solver IPOPT [24]. The initial solution was obtained in 74 iterations, where IPOPT ran for 46 s. The subsequent solution was acquired in 16 iterations and took 8.2 s. The drone successfully avoided all obstacles and flew from the initial to the target point. The path of the drone is shown in Fig. 7, where the blue spheres are obstacles, the green point is the start, and red is the position of the goal. The SoC and other trajectory states, together with the control trajectory, are shown in Fig. 8.

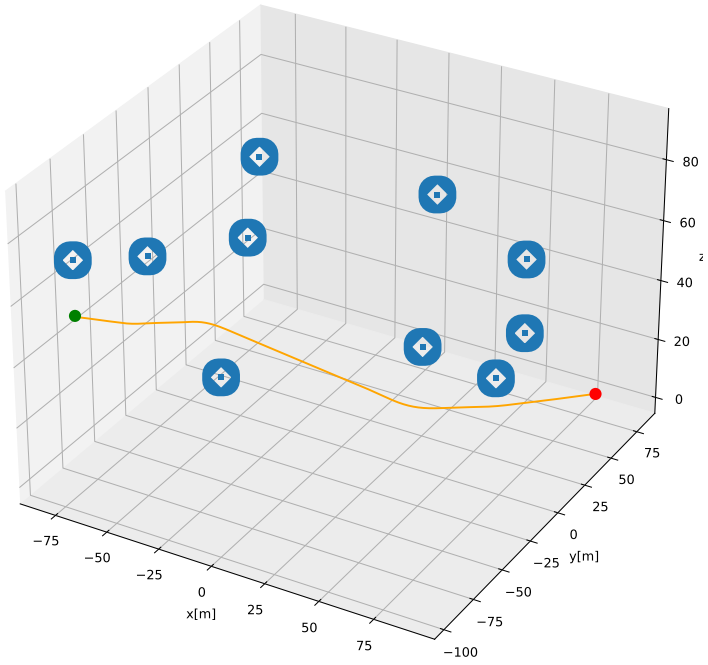
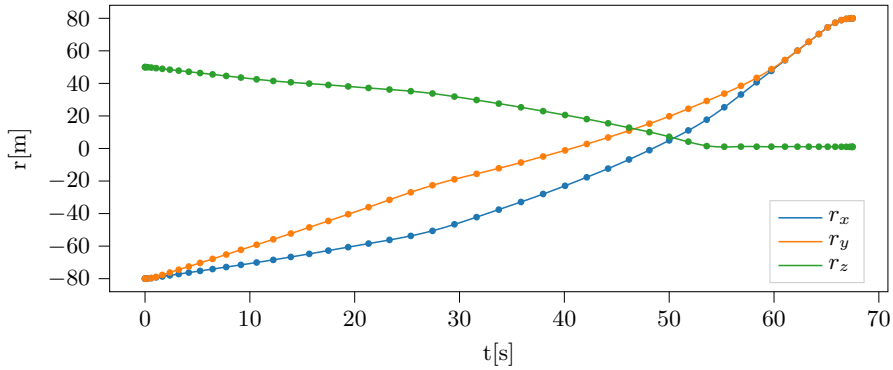
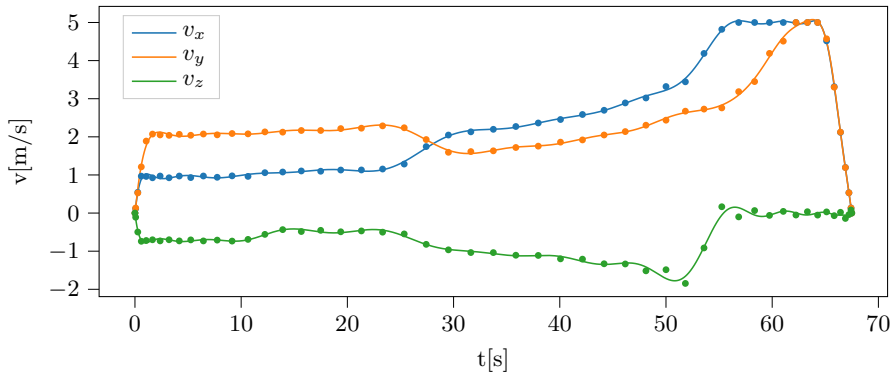


Fig. 7: Drone passage through an environment with obstacles.

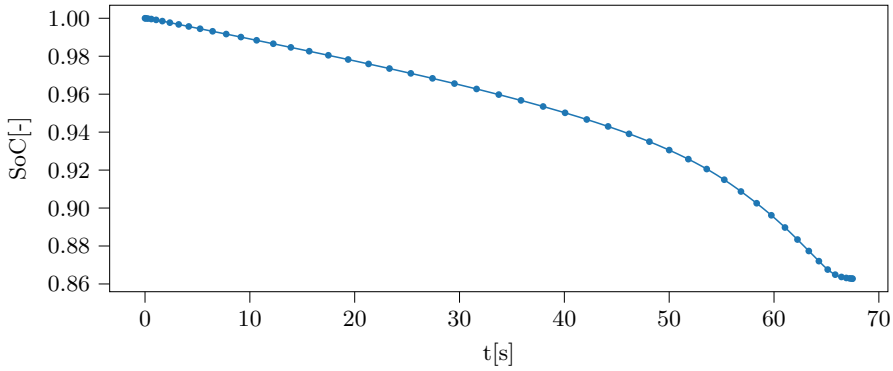
The results show that the optimal trajectory that considers battery discharge has been successfully found. The problem could be extended in the future, for example, to reflect the temperature during battery discharge, scheduling battery replacement at service stations or the interaction of several drones simultane-



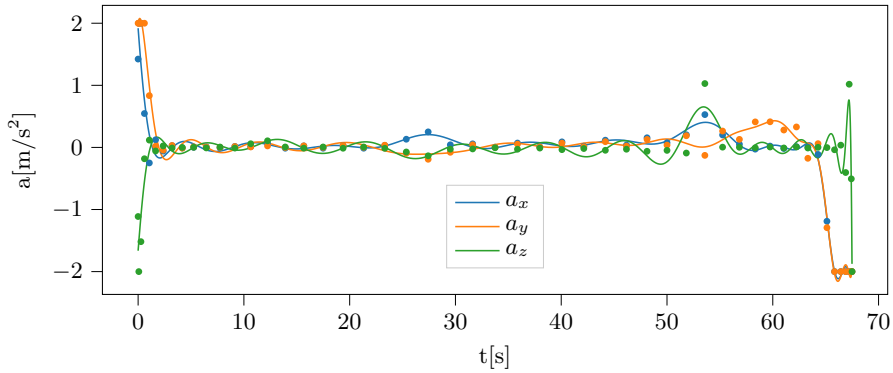
(a) Position



(b) Velocity



(c) State of Charge



(d) Acceleration

Fig. 8: Trajectory of state and control

ously. Furthermore, the PSM could be extended by adaptive mesh refinement, segmentation, or more accurate initialization.

5 Conclusion

The paper presented the DronePort system for smart drone management during long-term missions. First, the purpose of the system was outlined, and its most important parts were introduced. Afterward, the simulation environment was described, which will significantly help to make the whole development more efficient and at the same time reduce costs. Finally, the DronePort Traffic Control system for controlling and scheduling the swapping of drone batteries was introduced. An example of collision-free trajectory planning considering battery capacity using a Chebyshev pseudospectral optimal control was presented within the Traffic Control.

Acknowledgments

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826610. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Austria, Belgium, Czech Republic, France, Italy, Latvia, Netherlands.

References

1. H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges," *IEEE Access*, vol. 7, pp. 48 572–48 634, 2019.
2. D. Malyuta, C. Brommer, D. Hentzen, T. Stastny, R. Siegwart, and R. Brockers, "Long-duration fully autonomous operation of rotorcraft unmanned aerial systems for remote-sensing data acquisition," *Journal of Field Robotics*, vol. 37, no. 1, pp. 137–157, 2020.
3. N. K. Ure, G. Chowdhary, T. Toksoz, J. P. How, M. A. Vavrina, and J. Vian, "An automated battery management system to enable persistent missions with multiple aerial vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 1, pp. 275–286, 2015.
4. B. Michini, T. Toksoz, J. Redding, M. Michini, J. How, M. Vavrina, and J. Vian, "Automated Battery Swap and Recharge to Enable Persistent UAV Missions," in *Infotech@Aerospace 2011*. Reston, Virigina: American Institute of Aeronautics and Astronautics, mar 2011, pp. 0–10. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2011-1405
5. A. Koubaa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, and M. Khalgui, "Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey," *IEEE Access*, vol. 7, pp. 87 658–87 680, 2019.
6. N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

7. P. Castillo-Pizarro, T. V. Arredondo, and M. Torres-Torriti, "Introductory survey to open-source mobile robot simulation software," in *2010 Latin American Robotics Symposium and Intelligent Robotics Meeting*. IEEE, 2010, pp. 150–155.
8. A. Staranowicz and G. L. Mariottini, "A survey and comparison of commercial and open-source robotic simulator software," in *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments*, 2011, pp. 1–8.
9. L. Pitonakova, M. Giuliani, A. Pipe, and A. Winfield, "Feature and performance comparison of the v-rep, gazebo and argos robot simulators," in *Annual Conference Towards Autonomous Robotic Systems*. Springer, 2018, pp. 357–368.
10. E. Ebeid, M. Skriver, K. H. Terkildsen, K. Jensen, and U. P. Schultz, "A survey of open-source uav flight controllers and flight simulators," *Microprocessors and Microsystems*, vol. 61, pp. 11–20, 2018.
11. F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotors—a modular gazebo mav simulator framework," in *Robot Operating System (ROS)*. Springer, 2016, pp. 595–625.
12. G. Silano, P. Oppido, and L. Iannelli, "Software-in-the-loop simulation for improving flight control system design: a quadrotor case study," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2019, pp. 466–471.
13. S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, M. Hutter and R. Siegwart, Eds. Cham: Springer International Publishing, 2018, pp. 621–635.
14. Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Conference on Robot Learning*, 2020.
15. F. F. Monteiro, A. L. B. Vieira, J. M. X. N. Teixeira, V. Teichrieb *et al.*, "Simulating real robots in virtual environments using nvidia's isaac sdk," in *Anais Estendidos do XXI Simpósio de Realidade Virtual e Aumentada*. SBC, 2019, pp. 47–48.
16. E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1321–1326.
17. O. Michel, "Cyberbotics ltd. webots™: professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, 2004.
18. D. Zhang, S. Dey, H. E. Perez, and S. J. Moura, "Real-time capacity estimation of lithium-ion batteries utilizing thermal dynamics," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 992–1000, 2020.
19. D. E. Kirk, *Optimal Control Theory: An Introduction*. Dover Publications, 2004.
20. L. N. Trefethen, *Spectral Methods in MATLAB*. Society for Industrial and Applied Mathematics, jan 2000.
21. J. P. Boyd, *Chebyshev and Fourier Spectral Methods*, 2000.
22. I. M. Ross and M. Karpenko, "A review of pseudospectral optimal control: From theory to flight," *Annual Reviews in Control*, vol. 36, no. 2, pp. 182–197, dec 2012.
23. W. E. Hart, J.-P. Watson, and D. L. Woodruff, "Pyomo: modeling and solving mathematical programs in python," *Mathematical Programming Computation*, vol. 3, no. 3, pp. 219–260, 2011.
24. A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.