PAseos Simulates the Environment for Operating Multiple Spacecraft

Gómez, Pablo; Östman, Johan; Shreenath, Vinutha Magal; Meoni, Gabriele

DOI
10.1109/JSTARS.2024.3445506

Publication date
2024

Document Version
Final published version

Published in
IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing

PAseos Simulates the Environment for Operating Multiple Spacecraft

Pablo Gómez, Johan Östman, Vinutha Magal Shreenath, and Gabriele Meoni, *Member, IEEE*

***Abstract*—The next generation of spacecraft technology is anticipated to enable novel applications, including onboard processing, machine learning, and decentralized operational scenarios. Although several of these applications have been previously investigated, the real-world operational limitations associated with actual mission scenarios have been only superficially addressed. Here, we present an open-source Python module called PASEOS, capable of modeling operational scenarios involving one or multiple spacecraft. It considers several physical phenomena, including thermal, power, bandwidth, and communications constraints, and the impact of radiation on spacecraft. PASEOS can be run as a high-performance-oriented numerical simulation and/or in a real-time mode on edge hardware. We demonstrate these capabilities in three scenarios: one in real-time simulation on a Unibap iX-10 100 satellite processor, another in a simulation modeling an entire constellation performing tasks over several hours, and one training a machine learning model in a decentralized setting. While we demonstrate tasks in Earth orbit, PASEOS also allows deep space scenarios. Our results show that PASEOS can model the described scenarios efficiently and thus provide insight into operational considerations. We show this by measuring runtime and overhead as well as by investigating the constellation's modeled temperature, battery status, and communication windows. By running PASEOS on an actual satellite processor, we showcase how PASEOS can be directly included in hardware demonstrators for future missions. Overall, we provide the first solution to holistically model the physical constraints spacecraft encounter in space. The PASEOS module is available online with extensive documentation, enabling researchers to incorporate it into their studies quickly.**

***Index Terms*—Computational modeling, distributed computing, edge computing, onboard machine learning, spacecraft operations, satellite constellations.**

Manuscript received 19 December 2023; revised 12 July 2024; accepted 1 August 2024. Date of publication 2 September 2024; date of current version 2 October 2024. The work of Johan Östman was supported by Vinnova under Grant 2020-04825, and the work of Vinutha Magal Shreenath under Vinnova Grant 2021-03643. *(Corresponding author: Gabriele Meoni.)*

Pablo Gómez is with the Advanced Concepts Team, European Space Agency, European Space Research and Technology Centre, 2201 AZ Noordwijk, The Netherlands, and also with the AI Sweden, 417 56 Göteborg, Sweden (e-mail: pablo.gomez@esa.int).

Johan Östman and Vinutha Magal Shreenath are with the AI Sweden, 417 56 Göteborg, Sweden (e-mail: johan.ostman@ai.se; vinutha@ai.se).

Gabriele Meoni was with the AI Sweden, 417 56 Göteborg, Sweden, and also with the Department of Space Engineering, Faculty of Aerospace Engineering, TU Delft, 2629 HS Delft, The Netherlands. He is now with the Φ-Lab, European Space Agency, ESA Centre for Earth Observation, 00044 Frascati, Italy (e-mail: gabriele.meoni@esa.int).

This article has supplementary downloadable material available at https://doi.org/10.1109/JSTARS.2024.3445506, provided by the authors.

Digital Object Identifier 10.1109/JSTARS.2024.3445506

I. Introduction

THE last two decades have been characterized by significant changes in the availability, types, and usage of spacecraft, especially satellites in Earth orbit and beyond [1], [2]. Miniaturization has enabled operating satellites on the scale of centimeters with just a few kilograms of mass [3]. At the same time, the cost of launching satellites has plummeted, which opens up various new applications for both private and public actors [4]. Furthermore, with the advent of constellations for Earth observation (EO) and the increased availability of computation, tools to deal with, e.g., scale, complexity, and coordination of such distributed systems in space are required.

Many anticipated future applications are related to data processing and transmission [1], [5], also directly implemented on board satellites. Indeed, with the rise of artificial intelligence methods in virtually all domains, the next generation of satellites is likely to be equipped with novel tools and hardware [6] to enable machine learning methods and other technological innovations such as intersatellite links (ISLs) [7]. To this end, a growing corpus of research has been exploring novel applications suitable for this new paradigm, ranging from machine learning applications in remote sensing for EO applications [8], [9], [10], planetary exploration [11], [12] to federated learning in low-Earth orbit (LEO) constellations [13], [14]. However, there are fundamental constraints in the operations of LEO satellites related to, e.g., short communication windows with ground stations (resulting in large latencies and poor availability) [15], temperature, radiation, and power budgets [16]. In addition to a hostile environment, other challenges include increased hardware complexity and the absence of a communication network [17]. Naturally, such constraints extend well beyond LEO. Therefore, to facilitate resilient and realistic scenario planning, it is imperative to integrate these realistic constraints right in the algorithmic design and demonstration of space applications. Although specialized simulation tools are available for many facets of onboard operations—like communications [18], [19] or orbital dynamics [20]—there is no comprehensive simulation approach that encompasses the entire environment. This calls for a more unified and holistic simulation solution. Further, these tools are often incompatible with state-of-the-art machine learning and high-performance computing tools, such as the message passing interface (MPI) [21].

The primary purpose of this article is to fill this gap by introducing PAseos Simulates the Environment for Operating multiple Spacecraft (PASEOS), an open-source software

instrument explicitly engineered to accurately model the diverse constraints induced by the harsh space environment. PASEOS uses a modular design, thereby enabling the expansion of default functionalities. Furthermore, its compatibility with discrete-event and time-based simulations allows for synchronous and asynchronous operations, thereby broadening the horizon of potential use cases. It supports a variety of scenarios ranging from modeling individual satellites to large-scale constellations. PASEOS can operate like a classical numerical simulation, minimizing time-to-solution or in a real-time mode based on a clock, e.g., the system clock on which the hardware is running. PASEOS is implemented in Python and is entirely agnostic to any machine learning frameworks, such as PyTorch [22] or TensorFlow [23], or even the specific application. The utilization of Python and, thus, its support for machine learning frameworks such as PyTorch or TensorFlow, combined with its customization and modularity, render PASEOS particularly apt for early technology demonstrations related to machine learning. It enables easy initial consideration of various operational aspects for spacecraft with low computational cost and enables the simulation of a large number of spacecraft.

Designed to act as a cosimulator for onboard applications and decentralized missions, PASEOS distinguishes itself by being low overhead. Moreover, PASEOS is compatible with cutting-edge tools for high-performance computing, such as MPI facilitating simulations involving the cooperation of multiple spacecraft. In a concurrent work, these features have allowed considering various constraints during decentralized training of a machine learning model on multiple GPUs [24]. PASEOS thus introduces a complementary approach to existing simulation tools, which are typically designed for modeling individual satellites with specific use cases in mind that may not be relevant to innovative onboard applications. For instance, the commercial Ansys Software Tool Kit (STK) is well-suited for conducting high-fidelity simulations of spacecraft modeling synthetic aperture radar, infrared imaging, or radio-frequency communications [25]. Yet, due to the high fidelity, these simulations demand significant computational resources, and incorporating novel onboard applications, such as machine learning, into STK is not straightforward. Similarly, in a concurrent open-source project from the University of Tokyo, Spacecraft Simulation Environment(S2E), written in C++, presents a simulator for the onboard environment of spacecraft [26]. Released in December 2022, many of the S2E project's crucial components, like power consumption, temperature, and radiation, are still under development at the time of writing. Another option is the Java-based Virtual Satellite framework by DLR, enabling users to model the entire satellite lifecycle [27]. However, this framework does not target novel onboard applications such as distributed machine learning. Therefore, the introduction of PASEOS addresses a critical gap in the current landscape of spacecraft simulation and modeling.

This work describes the system design and mathematical modeling inside of PASEOS. Further, we demonstrate its capabilities on three applications. First, using actual satellite hardware (Unibap iX-10 100 satellite processor), we demonstrate real-time modeling of onboard detection of volcanic eruptions using a single satellite. Second, we model a large LEO constellation's operational behavior and constraints limited by thermal, power, and communication constraints. The final example demonstrates a fully decentralized machine learning application on two satellites modeled with PASEOS using heterogeneous data. PASEOS source code is open-source and available online, and we encourage modification and community contributions.[1] We hope to inspire and enable a broad variety of follow-up research by making PASEOS modular, user-friendly, and efficient. Thus, the contributions of this work are as follows:

1) Creation of the open-source Python framework PASEOS to model a broad range of physical aspects and constraints of spacecraft operations (e.g., thermal management, power management, radiation effects and others).
2) Demonstration that the framework can be used in a wide range of scenarios, including operations on actual satellite hardware for real-time implementations, large-scale modeling on supercomputers, and study distributed machine learning approaches.
3) Detailed analysis of the runtime overhead of PASEOS models in a scenario studying the detection of volcano eruptions on Unibap iX-10 100 processors.
4) Demonstration of the direct impact operational constraints can have on machine learning methods due to power budgets and communication windows.

II. Methods

This section describes the physical models implemented in PASEOS and some of the design considerations. In the following, any type of device, vehicle or similar, modeled in PASEOS, will be referred to as an *actor*.

A. Modeling of Spacecraft Constraints

Operating spacecraft poses vastly different challenges than operating any type of vehicle or asset on Earth [15], [28], [29]. Here, we briefly outline the constraints PASEOS considers and how they are modeled. Overall, PASEOS aims to strike a compromise between computational complexity and physical fidelity that enables it to be run in parallel to perform an operation, e.g., the training of a neural network, and still accounting for various constraints on different hardware devices, including embedded systems.

Overall, both physical constraints, such as thermal management or the availability of communication links, as well as operational constraints, such as per-user allocated time slots in missions, can be modeled with PASEOS. An overview of the modeled constraints can be found in Fig. 1, even customized constraints utilizing external software are possible. In this section, most constraints are explained in relation to orbits around Earth. They translate, however, to another scenario, such as orbits around other celestial bodies or deep space missions.

1) Astrodynamics Modeling: Many of the following constraints depend on the exact ephemerides, i.e., positions and

[1] [Online]. Available: https://github.com/aidotse/PASEOS. Last accessed on 19 December 2023.

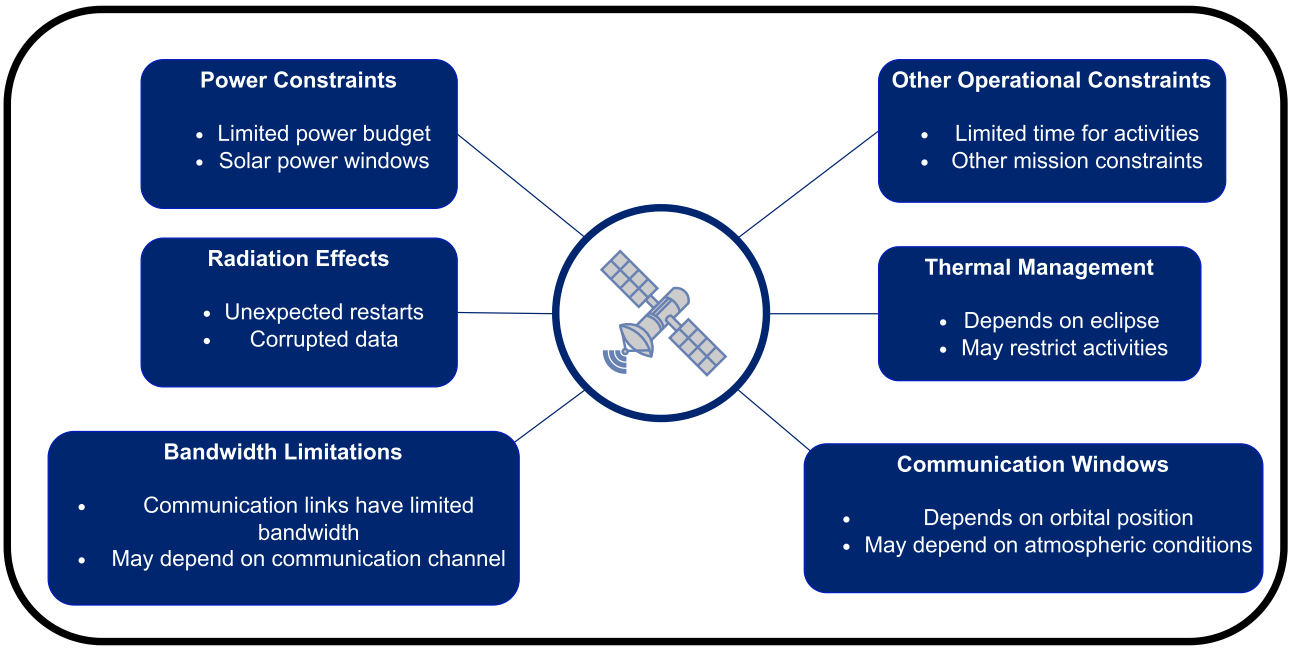

Fig. 1. Overview of the constraints modeled in PASEOS.

velocities, of the spacecraft and relevant celestial bodies, such as the Sun or the body the spacecraft may be orbiting. In PASEOS, we assume each actor to have a *central body* that it is gravitationally bound to, e.g., the Earth or the Sun. In its current Version 0.1.3, PASEOS uses pykep [30] to model Keplerian orbits around the central body. The JPL low-precision ephemerides[2] are used to model the position of the Earth and Sun. Even though we focus on simple Keplerian orbits in this work, one can employ more complex dynamics models ranging from propagators such as SGP4 [31] using two-line elements to high-fidelity modeling through *orekit* [20] as described in the documentation. All positions and velocities in PASEOS are modeled in the inertial frame of the central body.

2) Communication Windows: One of the central constraints in spacecraft operations, especially in LEO, is the communication windows during which a spacecraft can communicate with ground stations on Earth or with other spacecraft. In practice, these windows can depend on a large variety of factors such as the type of communication channel (e.g., optical, radio), environmental factors, such as atmospheric conditions, and others [32].

In PASEOS, we consider communications to be limited by visibility. For simplicity and computational efficiency, Version 0.1.3 considers only this constraint for communication between spacecraft (and ground stations). Thus, it is computed whether the sphere bounding the central body is obstructing the view between actors. For the availability of a ground link between a spacecraft and ground station, we utilize the Skyfield Python module [33] to compute the angle of a spacecraft above the horizon. Further, Skyfield is also used to account for ground stations' movement due to the Earth's rotation. Ground stations on other celestial bodies are not yet supported. Thus, based on these assumptions, PASEOS can be used to compute the availability and duration of communication windows between actors.

3) Data Transmission Rate and the Communication Channel: Aside from the availability of a communication link, a second factor for data transmission is related to the communication channel's effects, which affect the effective data transmission rate available for the user and impact the quality of data transmitted. Especially optical links from satellites in orbit around Earth to the ground are highly dependent on atmospheric conditions [32], [34]. Similarly, space weather events like coronal mass ejections or solar flares, e.g., can impact transmissions to Earth [35]. These effects can introduce errors in the communication, which could lead to retransmissions of data packets or require additional data redundancy, which impacts the effective downlink data rate available to the user. In addition, other factors such as the used transmission modulation, channel encoding strategy, and need for pilot synchronization play a significant role in the effective user transmission data rate [36].

However, high-fidelity modeling of these factors can be tremendously complex, requiring dedicated simulation software, such as *ns-3* or *OMNeT++* [18], [19]. PASEOS supports this via custom properties described in Section II-A8. Natively, PASEOS allows using an average data transmission rate to be considered for computing required transmission times. Such data transmission rate is regarded as constant and, within PASEOS, independent of the distance between the actors and the communication link.

4) Power Budgets: Another constraint relevant to all kinds of space missions is the available power budget and power systems [15], [37].

[2][Online]. Available: https://esa.github.io/pykep/documentation/planets.html#pykep.planet.jpl_lp. Last accessed on 19 December 2023.

Spacecraft typically rely on solar power and batteries near the Sun and inner planets. Further out, spacecraft rely on radioisotope thermoelectric generators [38]. Land-based assets such as rovers typically also rely on either of these methods.

We assume spacecraft to have a battery with a fixed capacity. For simplicity, we model solar power generation in PASEOS as an increase in the battery's state of charge (SoC) at a constant rate while a spacecraft is not in the eclipse of the central body relative to the Sun. The power budget is currently not considered for ground-based actors as we are mainly focusing on Earth-based ground segments. However, modeling of ground-based assets' power budgets can follow analogously to spacecraft if, e.g., rover operations are modeled.

The SoC is reduced through the spacecraft activities, which—in the context of PASEOS—are user-defined operations that consume charge at a constant rate during operations.

5) Thermal Management: Another critical constraint on the activities a spacecraft can perform is posed by managing the spacecraft's temperature, both in terms of low and high temperatures. Spacecraft hardware will typically have a limited operational and survival range in terms of temperatures [16]. Hence, such constraints must be respected during any operations the spacecraft performs. Due to the surrounding near-vacuum, heat dissipation is typically a major concern. However, low temperatures can also pose problems, especially when the spacecraft is in the central body's eclipse or deep space. Therefore, it is necessary to rely on a thermal model to anticipate and prevent violation of these constraints.

In PASEOS, we rely on a formulation similar to the one described by Martínez [39]. We model the change in the spacecraft's temperature T as

$$mc\frac{\Delta T}{\Delta t} = \delta_{\mathrm{a}}\dot{Q}_{\mathrm{solar}} + \delta_{\mathrm{a}}\dot{Q}_{\mathrm{albedo}} + \dot{Q}_{\mathrm{IR}} + \dot{Q}_{\mathrm{activity}} - \dot{Q}_{\mathrm{diss}} \quad (1)$$

where m is the spacecraft mass, c its thermal capacity, $\delta_{\mathrm{a}} = 1$ if the spacecraft is in eclipse and 0 otherwise and the $\dot{Q}$ are heat fluxes. In particular, the individual heat fluxes are given as

$$\dot{Q}_{\mathrm{solar}} = \alpha_{\mathrm{a}}\, E_{\mathrm{s}}\, A_{\mathrm{a,Sun}} \quad (2)$$

where $\dot{Q}_{\mathrm{solar}}$ is the radiative flux produced by the sun, $\alpha_s \in [0, 1]$ is the spacecraft solar absorptance, $A_{\mathrm{s,Sun}}$ is the area facing the Sun, and E_{s} the solar irradiance

$$\dot{Q}_{\mathrm{albedo}} = 0.5\, \alpha_{\mathrm{a}}\, \rho_{\mathrm{b}}\, E_{\mathrm{s}}\, A_{\mathrm{a,albedo}} \quad (3)$$

where $\dot{Q}_{\mathrm{albedo}}$ is the sun heat flux generated back-scattered by Earth, $A_{\mathrm{a,albedo}}$ being the area facing the albedo, and ρ_{b} being the central body's reflectance. The 0.5 factor stems from a simplification as we currently do not compute angles between the Sun, the central body and spacecraft to reduce computational costs

$$\dot{Q}_{\mathrm{IR}} = \frac{r_{\mathrm{b}}^2\, \epsilon_{\mathrm{a}}\, \epsilon_{\mathrm{b}}\, \sigma\, T_{\mathrm{b}}^4\, A_{\mathrm{a,b}}}{\bar{r}_{\mathrm{a}}^2} \quad (4)$$

where $\dot{Q}_{\mathrm{IR}}$ is the radiative flux due to the Earth black body emission, $A_{\mathrm{a,b}}$ is the spacecraft area facing the central body, $\bar{r}_{\mathrm{a}}$ is the spacecraft's distance to the central body center, $\epsilon_{\mathrm{a}} \in [0, 1]$ the spacecraft's infrared emissivity (i.e., absorptance), σ the Boltzmann constant, and r_{b}, ϵ_{b}, and T_{b} are the central body's radius, infrared emissivity, and temperature. Furthermore,

$$\dot{Q}_{\mathrm{activity}} = \kappa P_{\mathrm{A}} \quad (5)$$

where $\dot{Q}_{\mathrm{activity}}$ is the heat flux from the spacecraft hardware, P_{A} is the power consumption rate of an activity A, and κ is a user-defined parameter describing conversion rate into heat. Finally,

$$\dot{Q}_{\mathrm{diss}} = \epsilon_{\mathrm{b}}\, A_{\mathrm{b}}\, \sigma T^4 \quad (6)$$

where $\dot{Q}_{\mathrm{diss}}$ is the heat emitted from the spacecraft. While PASEOS does not enforce consideration of the spacecraft temperature (except disallowing temperatures below 0K), it enables the user to query the current temperature and formulate abort conditions based on it.

6) Radiation Effects: Another constraint and physical factor to consider are the effects of radiation, especially beyond LEO, where spacecraft are still fairly protected by the Earth's magnetic field [40], [41]. In practice, these events can lead to data corruption, software faults, or permanent hardware damage [6]. In PASEOS, we model three different types of effects on operations due to single event effects [15]:

1) *Data corruption* with a certain probability due to single event upsets. For that, PASEOS models flipped bits that occur according to a Poisson distribution with a rate r_{d}.
2) *Unexpected software faults* leading to a random interruption of activities following a Poisson distribution with rate r_{i}.
3) *Device failures* following a Poisson-distribution with rate r_{f}, which can be imputed mostly to single event latch-ups.

Given the dependence on spacecraft-specific hardware and orbit, the definitions of these rates are left to the user. We provide an example of modeling for effects such as total ionization dose (TID) in the documentation and Section II-A8.

7) Operational Constraints: In addition to the constraints imposed due to physics and the space environment, missions typically have many objectives, and there are various stakeholders for any spacecraft [42]. Therefore, there are often operational constraints to be considered that are imposed through the mission profile. PASEOS enables these by allowing user-defined constraints based on arbitrary parameters that are evaluated during activities and lead to interruption of the activity if the constraints are not met. This gives users a broad range of possibilities, from imposing strict hardware limits, such as respecting a minimum SoC or specific temperature range, to requiring time limits or factors outside the PASEOS simulation.

8) Custom Properties and Integrated External Software: Finally, beyond the natively integrated physical models, PASEOS provides the capability to integrate external software. The physical models presented before are designed to be lightweight and computationally feasible on embedded devices. To this aim, the physical models do not provide the same fidelity as dedicated simulators such as STK. To reduce the reality gap, the users can rely on external software for individual modeling as needed and directly wrap it in PASEOS to utilize more complex but accurate physical models. This is demonstrated in the examples

of PASEOS,[3] where modeling additional phenomena, more accurate dynamics using *orekit* are shown.

PASEOS supports the implementation of this through so-called *custom properties* and the option to define custom propagators. A custom property is an additional physical quantity of an actor that is updated in the PASEOS main simulation loop—as the other physical quantities—through a user-defined function. As specified in Section IV-D, we provided an example using custom properties to model TID effects. Fundamentally, this allows the integration of a large variety of modeling and other software.

B. Software Design

In the following section, we briefly elaborate on the design philosophy of PASEOS with regard to user interaction and validation of the models. Even though we already provide concrete application results in this work, PASEOS aims to enable various future applications; consequently, a flexible and generic design is paramount.

1) Actors: As an abstraction of the variety of different assets available on the ground and in space, we simulate them in PASEOS as so-called *actors*. Fundamentally, we distinguish between ground-based actors and spacecraft actors. In the current version, the main feature associated with the former is modeling the change in position due to the rotation of the central body. For spacecraft actors, a variety of models can be enabled describing all the physical and operational aspects described in Section II-A. Initially, one only needs to define the type of actor, a name for it, and the current local time of the actor. Depending on the desired models, additional parameters, e.g., position and velocity for orbits, need to be specified.

2) Activities: Activities are the second central abstraction in PASEOS. They serve to describe any operation the user wants to model. For example, we may want our spacecraft actor to capture data with one of its sensors and process that data. PASEOS allows the specification of an (asynchronous) function that is executed in the background while PASEOS models the physical constraints. Further, a constraint function, which is then evaluated repeatedly at a fixed timestep, can be used to interrupt an activity. For each activity, one has to specify the power consumption to allow PASEOS to model excess heat and the change in the battery's SoC.

Users can either let PASEOS run operations asynchronously to model updates or run operations and then advance PASEOS' simulation time.

3) Discrete Event Versus Time Based: Another challenge for PASEOS is that network-oriented simulations, such as *ns-3* [19], typically focus on providing a discrete-events simulation, allowing users to skip to the next event. However, physical simulations modeling ordinary differential equations, such as the one in (1), however, are usually solved in a time-based fashion with discrete time-stepping schemes [43]. In PASEOS, these two simulation paradigms meet as both kinds of simulations are addressed and

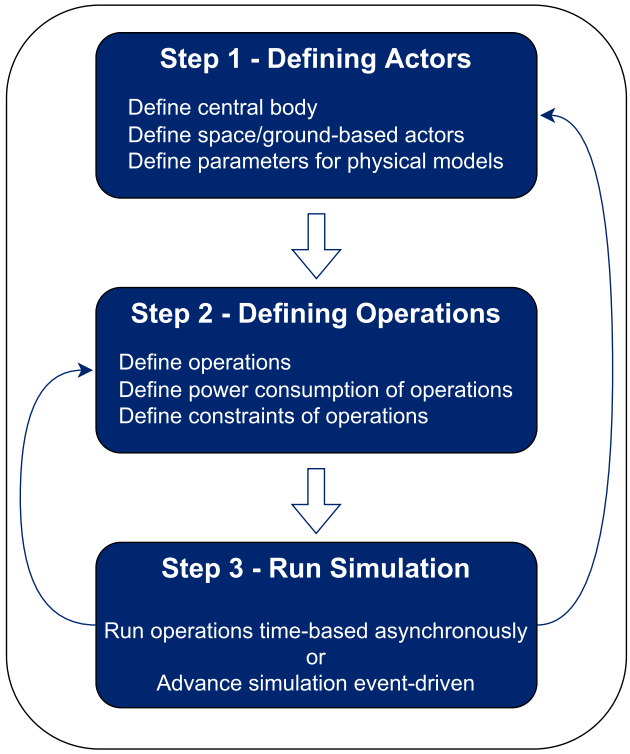

Fig. 2. Overview of the workflow using PASEOS.

part of PASEOS. PASEOS operates at the intersection of both, performing numerical simulations of physical processes while modeling discrete events such as communication windows becoming available. One can choose a fully real-time operation of PASEOS, where it will run the physical models asynchronously while performing a user-defined activity. Alternatively, one can manually ask PASEOS to advance its simulation state to a specific time interrupted by potential events of interest to the user.

4) Using PASEOS: With the relevant terminology introduced, we can describe the main workflow of PASEOS. Fig. 2 gives an overview of the high-level use of PASEOS. It fundamentally requires two definition steps where, first, the actors are modeled and, second, the modeled operations are defined. Operations can then be performed repeatedly in either a discrete-event or time-based fashion. At any point in the simulation, the user may benefit from detailed output logs on actor status and activities (written to a *.csv file) and/or from visualizations.

5) Validation: Given the multitude of physical models in PASEOS and the complex interaction due to asynchronously running activities, a thorough validation of the software is both critical and challenging. Test-driven development has been employed to design many of the individual components, and no contributions are merged without code review, appropriate tests, and passed automated tests. Furthermore, we rely on a comprehensive suite of automated tests using *pytest*[4] to validate the continued correctness of the implemented models. In

[3][Online]. Available: https://github.com/aidotse/PASEOS/tree/main/examples. Last accessed on 19 December 2023.

[4]pytest: helps you write better programs. Available online at: "https://docs.pytest.org/en/7.4.x/". Last accessed on 10 December 2023.

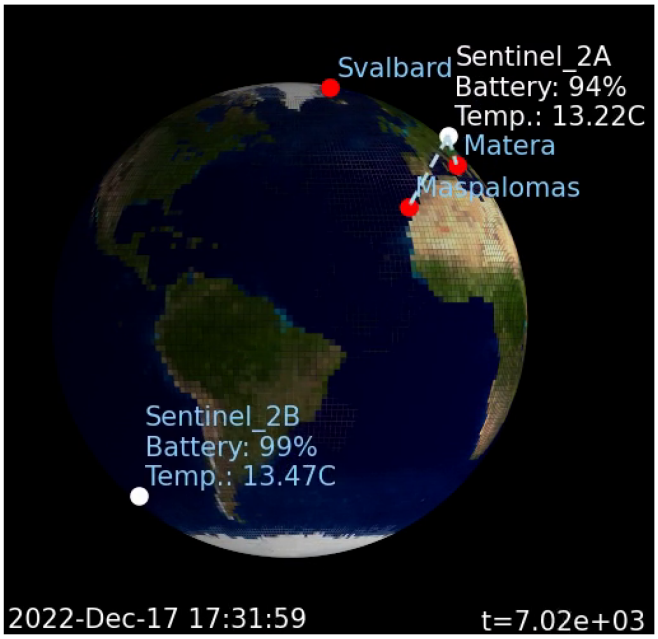

Fig. 3. PASEOS visualization showing the Sentinel-2 mission use case.

particular, a dedicated test is included for each physical model and functionality implemented in PASEOS, including checks of the correct implementations of the event-based and activities mode, constraints functions, and custom properties. To test the capability of PASEOS to assess if two actors are in the line of sight, we simulated different scenarios, including one satellite and a correspondent ground station or various satellites in controlled scenarios. For what concerns the communication window, we perform their estimation by leveraging *Skyfield* [33], a well-established tool, to check the visibility of the spacecraft from the ground. To validate our results, automated tests measure the available window length for the Sentinel-2 mission for the ground station in Maspalomas, Spain, and compare it with the values reported in [44]. The list of dedicated tests for each physical model or software feature to validate, a description and the expected reference value are displayed in supplementary materials.

The validation of the visualization utilities was performed through visual inspection. Fig. 3 showcases a frame obtained from the visualization toolkit when the setup of the Sentinel-2 mission is simulated in PASEOS. In particular, Fig. 3 shows the Sentinel-2A and Sentinel-2B satellites and the three mission ground stations. The dashed line between Sentinel-2A and the Matera and Maspolamas ground stations shows they are in line of sight.

Finally, the model of the operational constraints affecting the operations of Unibap iX-10 100 device was performed with the support of Unibap technical experts. A modular design approach enables the testing of individual components, such as the thermal model or communication links. Further, PASEOS is available online as an open-source software to enable anyone to inspect the code, provide feedback, and report bugs.

C. Modeling Asynchronous Operations Between Multiple Spacecraft

The PASEOS framework may be readily used to model operations involving multiple actors where each physically modeled actor contains a PASEOS instance. Note that this setting requires a change of perspective from Fig. 2, where there is a single actor of interest, to a system-level view where each actor is to be modeled. PASEOS does not aim to facilitate coordination, scheduling of activities, or other tasks specific to the modeled application. Instead, it aims to allow the user to model different scenarios, such as intermittent knowledge of other actors and centralized (e.g., federated learning) and decentralized (e.g., decentralized learning) operations. Each actor is assumed to operate independently, and any information about other actors should be acquired during the operation. Operations can either be performed independently, where the user takes the responsibility to advance the time of the PASEOS simulation manually, or they can operate in an asynchronous fashion with event-based activities.

Flexibility is an important design goal of PASEOS, and the user can equip a PASEOS instance with arbitrary capabilities by registering activities, see Section II-B2, consisting of the actions to be performed, the power consumption and constraints that must be satisfied for the activity to be performed. The action and constraint of an activity are coroutines, i.e., asynchronous functions, and rely on the *asyncio* Python module.[5] When an activity is performed, its action and constraint function are submitted to an event loop for execution. Meanwhile, PASEOS will monitor and update the internals of the actor, e.g., temperature and SoC, and advance the time. Note that PASEOS allows only one activity to be performed simultaneously. Given the arbitrary code execution inside the activity, a user can, however, perform different tasks within an activity, e.g., based on the actors' state.

To allow for the simulation of multiple spacecraft, the ability for the actors to interact is imperative. In PASEOS, communications are achieved by encapsulating transmission and reception as activities. Such activities should be designed to comply with the communication device for the given actor, see Section II-A3, for the communication windows to be properly calculated. For simulation on a single machine, communications may be emulated by simply imposing a delay (and power consumption) in the corresponding activity.

To use PASEOS directly on edge devices, e.g., spacecraft hardware, communication activities may be based on packages such as *gRPC* and *zeroMQ* to allow interaction over a network. Time synchronization between the actors is paramount for correct operation when working with edge devices. As the actors in PASEOS operate independently, they must make themselves known to others by, e.g., sending out heartbeats at given intervals containing the id of the actor and, optionally, metadata, e.g., position and velocity. An actor may also be unavailable due to low SoC, no line-of-sight connection, or a device failure. The absence of heartbeats entails the unavailability and other actors may respond accordingly. PASEOS does not automatically send

[5]asyncio – Asynchronous I/O. Available online at: "https://docs.python.org/3/library/asyncio.html". Last accessed on 19 December 2023.

heartbeats or check these factors but provides an API to serialize actors for network transmission and the user to tell each actor about its known and available peers. Thus, arbitrarily simple or complex connectivity constraints can be imposed.

If one wants to perform specific operations in a synchronized manner, it is necessary to synchronize actors in time by, e.g., providing a start time for an activity. Alternatively, time synchronization can be achieved by means of communication [45]. For example, one may utilize the Network Time Protocol [46] that relies on a master clock or decentralized approaches that rely on, e.g., the heartbeats [47].

III. Results

This section demonstrates the usage of PASEOS for three distinct modeling scenarios: EO, constellation design, and decentralized machine learning. The purpose is to illustrate the broad range of applications readily modeled with PASEOS's aid.

A. Single Actor: Onboard Volcano Detection Onboard Sentinel-2

This experiment showcases how one can use PASEOS in real-time simulations to emulate satellite onboard-processing scenarios on actual space hardware.

1) Setup: We design an experiment in which we register an activity to process satellite payload data and utilize a constraint function to check the availability of a link to transmit data to the ground. During the simulation, we profiled our code on a Unibap iX-10 100 satellite processor [6] to evaluate the overhead of PASEOS, i.e., the time spent to update the physical models compared to the time required for checking constraints and running user activities. In particular, we consider onboard detection of volcanic eruptions applied to Sentinel-2 L1-C, where the acquired data are processed directly onboard a satellite to spot possible volcanic eruptions and deliver early alerts to ground [9], [48].

To set up the orbit of the spacecraft actor, we used the ephemerides of the Sentinel-2B satellite at 2022-10-27T12:30:00Z, which were calculated using two-line elements. Because of that, the orbit of our actor is sun-synchronous. To set up the ground station actor position, we used the European Space Agency ground station, located at Maspalomas, Gran Canaria (27.7629° latitude, -15.6338° longitude at an elevation of 205.1 m). The ground station link is available if the satellite is 5° above the horizon. To check for the possibility of communicating, we used a constraint function that interrupts the user activity (i.e., the volcanic eruption detection) when the satellite is in line of sight (LOS) with the ground station.

To have sufficient energy to process data onboard, the actor is equipped with a 0.162 MJ battery, with a SoC of 1.0 at the beginning of the simulation. We assume a charging rate of 10 W. Since we model only one activity, i.e., detecting volcanic eruptions, we computed the charging rate needed to compensate for the power consumption due to that activity over the orbit (i.e., 10 W). This power consumption was estimated by matching the typical power budget per orbit for CubeSats ($\sim$6U) for the payload acquisition/processing chain performing data-processing on board (e.g., Φ-Sat-1[6]). By doing that, we are not modeling all components of the power budget needed to perform platform-level activities (e.g., attitude determination and control, thermal control, etc.), which are out of scope for this particular study but only the ones affecting this operation.

Furthermore, to increase the computational cost due to the update of the physical models in PASEOS and test the system in the worst case, we also equipped the space actor with a thermal and radiation model to measure their run time impact. In particular, to avoid the effects of radiation but track computational cost, we set up the data corruption rate, restart, and device failure to be 0—this does not affect the computational cost.

The simulation data consist of three Sentinel-2 L1C post-processed products showing volcanic eruptions of Etna (2021-08-30), La Palma (2021-09-30), and Mayon (2018-02-09), provided by Meoni et al. [49]. Each image is produced by cutting and mosaicing the 20 m bands B8A (Near-Infrared) and the B11 and B12 (Short-Wave Infrared) of multiple tiles over the area of the band B8A of the correspondent Sentinel-2 Raw granule [49]. We artificially extended the simulation time by repeatedly evaluating the images until we got 100 images in total for the simulation. In addition, we assumed all data to be already acquired and available for processing to disregard hardware-related delays that would occur in reality as we focus here on profiling PASEOS.

Code profiling was performed by using the Python module *yappi*.[7] In particular, we measured the central processing unit (CPU) time on the iX-10 100 device for three different values (0.25 s, 0.5 s, 1 s) of the *PASEOS timestep*, i.e., the interval time for updating PASEOS, its physical models, and checking user constraints. We performed three runs for each choice of PASEOS timestep. During each run, we measured the CPU time for the user activity to check user constraints (i.e., check of LOS with the ground station), the time to update PASEOS overall, and the individual times spent to update the radiation, the thermal, and the battery charge models. Results were averaged over the three runs for each test case. All the tests were performed sequentially with two warm-up runs with a PASEOS timestep of 1 s.

2) Onboard Volcanic Eruption Detection: The detection is based on a simplified implementation of the algorithm [50], presented in [49], which produces a bounding box surrounding the detected volcanic eruption and the associated geographical coordinates. Since the aim of the activity is to provide an early alert to promptly notify and locate a detected volcanic eruption, a possible alert message will provide the coordinates of the bounding box center or top-left/bottom-right points. One example of a detected volcanic eruption on the island La Palma is shown in Fig. 4.

3) Results of Profiling: Table I shows the simulation results for the runs on the iX-10 100 device. The activity was interrupted in each test after roughly 29 s of CPU time because the satellite

[6]PhiSat-1Nanosatellite Mission: https://www.eoportal.org/satellite-missions/phisat-1#hyperscout-2. Last accessed on 10 July 2023.

[7]Yet Another Python Profiler (yappi). Available online at: "https://pypi.org/project/yappi/." Last accessed on 19 December 2023.

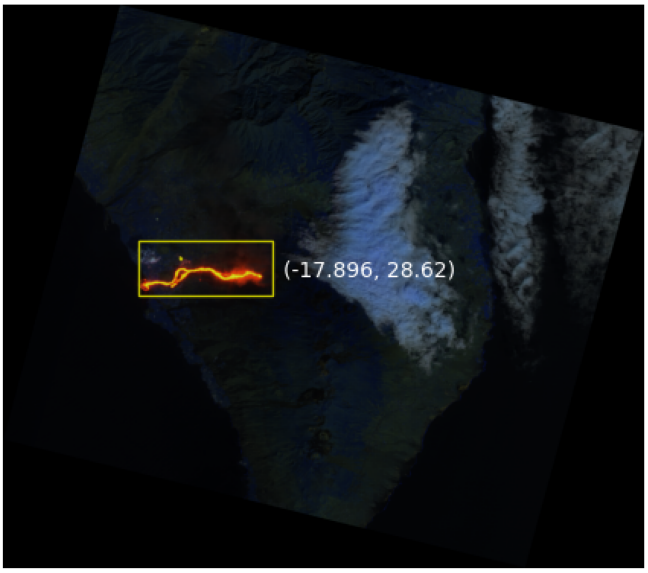

Fig. 4. Detected volcanic eruption on the La Palma (Spain) on 30 September 2021. The displayed coordinates (longitude, latitude) correspond to the center of the detected bounding box.

TABLE I
RESULTS OF PROFILING ON A UNIBAP IX-10 100 DEVICE FOR DIFFERENT PASEOS TIMESTEPS

All in [s](%)		PASEOS timestep		
Type		0.25	0.5	1.0
User-defined	User Activity	28.64 (97.47)	28.59 (98.66)	28.52 (99.27)
	LOS Constraint	0.29 (0.98)	0.14 (0.50)	0.072 (0.25)
PASEOS State Update	SoC	0.11 (0.39)	0.062 (0.21)	0.034 (0.12)
	Radiation	0.0086 (0.029)	0.0044 (0.015)	0.0024 (0.008)
	Thermal	0.11 (0.39	0.062 (0.21)	0.034 (0.12)
	Other	0.19 (0.65)	0.10 (0.34)	0.056 (0.19)
	Subtotal	0.43 (1.45)	0.23 (0.79)	0.13 (0.44)
Total		29.39	28.98	28.74

Each column reports the CPU time in seconds spent for the user-defined activity to update the physical models of PASEOS and their individual times. Percentages with respect to the total time are reported in brackets. The last column provides the time spent for the check of the LOS.

was found in LOS with the ground station. Indeed, the time to perform the user activity is similar for each value of the PASEOS timestep, while the time to update the PASEOS state and to check the user constraints grows almost linearly. However, even for the smallest PASEOS timestep, the latter are 0.43 s and 0.29 s, respectively, compared to 28.64 s for the user activity. Overall, PASEOS requires around 1.45% of the total run time. This demonstrates that PASEOS is a lightweight solution suitable for onboard processing use cases on embedded hardware.

As shown in Table I, the time for modeling battery charge is equal to the one for updating the thermal model. This is because the update of both the power and the thermal models requires checking whether the spacecraft actor is in eclipse. This operation is carried out only once for both models, and its run time is attributed equally to the thermal and power models. All the other operations to update the battery SoC and thermal models are negligible.

B. Multiple Actors: Communications Modeling of a Constellation

The capabilities of PASEOS to model the operational constraints of managing a constellation are demonstrated in this test case. Detailed results on the constellation's status over time are given, and a simple scaling study is performed to investigate PASEOS scaling abilities to showcase the potential of modeling large constellations.

1) Setup: The scenario investigated here is a LEO constellation consisting of 16 spacecraft in a Walker pattern [51] in four planes at 550 km altitude with an inclination of 10°. Operations of the constellation are modeled for 8 hours—that equals slightly more than five revolutions. The satellites are presumed to be equally equipped with a 1 MJ battery initially at randomly uniform SoC between 0.1 and 1.0. Each satellite is equipped with solar panels charging at 50 W. Satellites are assumed to have a mass of 50 kg, to be at 273.15 K initially and have absorptance of solar and infrared light of 1.0. Parameters are inspired by current large CubeSat platforms such as the EnduroSat *12U XL CubeSat Platform*. Thermal parameters were inspired by the work underlying the thermal model [39]. The areas facing the Sun and Earth are assumed to each be 2 m ^{2}. The emissive (radiating) area is presumed to be 4 m^2. The thermal capacity is assumed to be 1000 Jkg^{-1} K^{-1}. We assume half of the used wattage for satellite operations to be converted to heat. The Earth's temperature is assumed to be 288 K, its infrared emissivity at 0.6, and its solar reflectance at 0.3. Solar irradiance is estimated at 1360 W. The parameters are chosen to be in a typical range and do not represent a specific currently operational satellite.

Satellites have two operational modes: a standby mode called *Standby*, where they consume only 2 W and the other one is called *Processing*, where they consume 100 W. The satellites automatically switch to *Standby* if their battery's SoC falls below 0.2 or their temperature above 330 K.

In addition to the constellation, we monitor the availability of communication links to a satellite in geosynchronous orbit called *GeoSat* and the European Space Agency ground station at Maspalomas, Gran Canaria. As for the single satellite scenario, the ground station link is available if satellites are 5° above the horizon. The geosynchronous satellite is reachable from the Maspalomas station.

We use a time step of 1 s in PASEOS for the thermal and power models. Satellites decide every 600 s whether they are ready for *Processing*. If the constraints for *Processing* are violated during the 600 s interval, they switch to *Standby* for the remainder of the interval.

2) Constellation Analysis: We analyze the constellation regarding several operational factors. First off, in terms of time spent processing. As displayed in Fig. 5, roughly half of the satellites are processing at any time, and both battery SoC and temperature limit the periods of operation. Given the circular

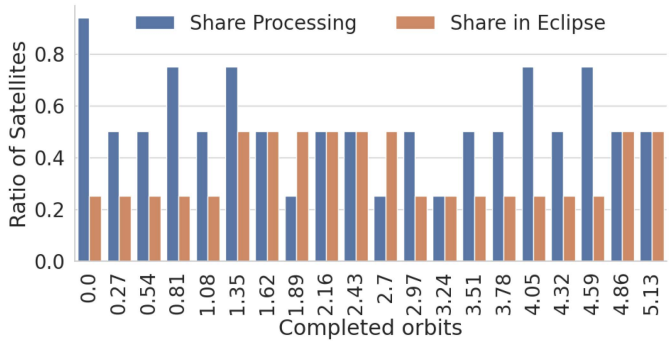

Fig. 5. Overview of the satellites able to process and in eclipse over time.

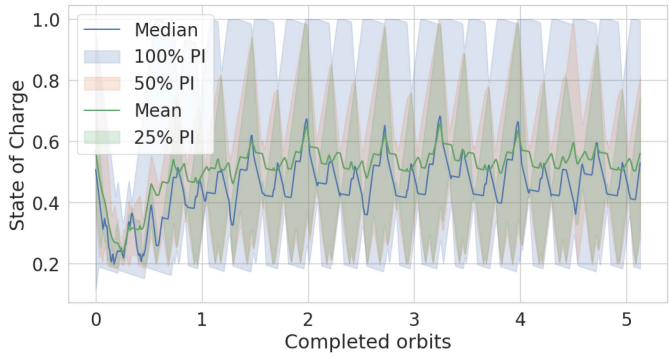

Fig. 6. Battery SoC of the constellation. Different colors indicate percentage intervals around the median.

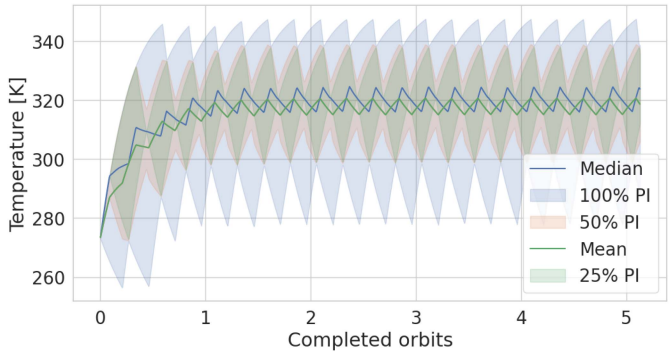

Fig. 7. Temperature of the constellation. Different colors indicate percentage intervals around the median.

LEO orbits of the constellation , the satellites spend a considerable time in eclipse, with 25% to 50% of the constellation being in eclipse at any moment. This also influences the operational temperature—as seen in Fig. 7–of the satellites, which rises quickly at the beginning when a large share begins processing, and the temperature falls, especially during eclipse.

Even though PASEOS' SoC model is simplistic, for now, complex power budget dynamics can be observed in Fig. 6. Satellites in the constellation fluctuate between a SoC of 0.2 and 1.0. The low standby consumption of the *Standby* activity means they never risk reaching critical SoCs. Overall, the constellation's behavior regarding satellites performing *Processing*

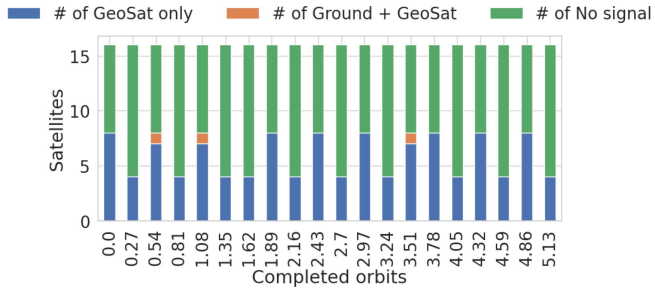

Fig. 8. Overview of the satellites' communication status over time.

and the constellation's temperature and SoC become stable and cyclic after roughly one orbital revolution.

In terms of communication status, 50% to 75% of the constellations are not within LOS of either the ground station or satellite in geosynchronous orbit, as can be seen in Fig. 8. As the geosynchronous satellite is reachable from Maspalomas, it is also in LOS of a constellation satellite when the ground station is.

3) Performance and Scaling: Running this scenario on an AMD Ryzen-5 consumer-grade CPU on a single thread requires about 200 s. Time is spent almost exclusively on the physical models and LOS checks. With 16 satellites, running the simulation for a simulation time of 600 s requires 8.3 s, 0.52 s per satellite. We investigated the scaling by increasing the number of satellites per plane. With a constellation of size 32, 16.4 s were necessary for 600 s simulation time, 0.51 s per satellite. The better-than-linear scaling is likely due to a constant initialization overhead in starting the simulation. With 512 satellites, the 600 s required 255.9 s computation time or 0.50 s per satellite. Thus, it is clear that PASEOS scales well even to a large number of satellites. Parallelization of these simulations is also trivial given the fully asynchronous nature of PASEOS that permits parallelization without any concern for shared memory of similar.

C. Onboard Machine Learning in Orbit

This test demonstrates how PASEOS can be employed to model and monitor constraints when solving a machine learning task in a decentralized setting. Limited communication windows, heterogeneous data, and power constraints are imposed while successfully solving a classification task.

1) Setup: The operational scenario in this example consists of two satellites in circular LEO at an altitude of 550 km with an inclination of 98.62°. They move in opposing directions and thus have only brief communication windows among them twice per orbital revolution. Both are equipped with a 0.1 MJ battery with an initial SoC randomly chosen between 0.6 and 0.8. They have solar panels which charge the battery at 50 W when not in eclipse. They are also equipped with intersatellite links, enabling them to transmit 1 Mbit/s among themselves when in LOS. The satellites are assumed to not communicate during the first ten orbital revolutions.

The satellites are tasked with jointly learning a binary classification task identifying an inner and outer circle by leveraging

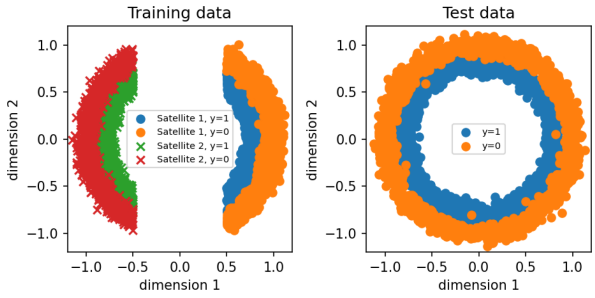

Fig. 9. Training and test data for the binary classification task as well as their distribution among satellites. Note the heterogeneity of the data distribution in the training set. $y = 0$ and $y = 1$ are the classes of the problem.

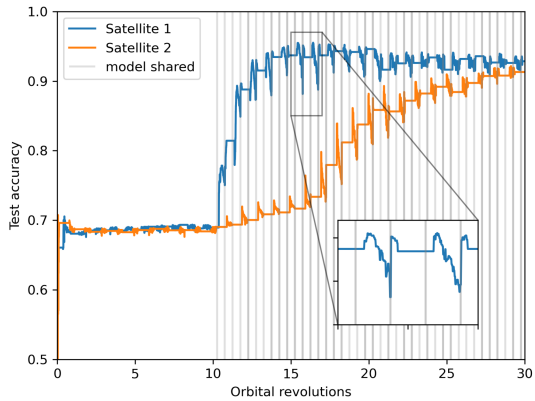

Fig. 10. Test accuracy over 30 orbital revolutions. Vertical gray lines indicate communication between the satellites. Constant accuracy stems from the battery SoC being below 0.5.

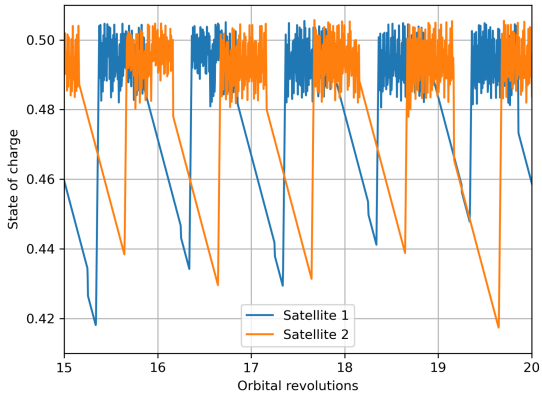

Fig. 11. SoC between orbital revolution 15 and 20.

data uniquely available to each satellite. The used dataset and its distribution among the satellites are depicted in Fig. 9. Notably, the distributions are heterogeneous, with the satellites only having access to data points with values in the first dimension above 0.5 and below -0.5, respectively. The test dataset is identical on both satellites and covers the complete feature space. A total of 4166 training samples are used, and the test dataset consists of 3300 samples. A two-layer dense neural network with ten neurons is trained using stochastic gradient descent with a learning rate of 0.1 and a binary cross-entropy loss function. Training is modeled for a total of 30 orbital revolutions. In each revolution, if allowed, the satellites communicate twice and will train an aggregated model in the window between communications. On average, Satellite 1 can train 42.3 epochs and 11.6 epochs in the two different windows, whereas Satellite 2 is able to train 4.1 epochs and 50.5 epochs. During operations, the satellites have three distinct operational modes they perform in descending priority:

1) *Standby*—When the SoC is below 0.5, the satellites stand by to conserve and charge their batteries. This mode drains 2 W.
2) *Model sharing*—When the satellites are in LOS of each other they exchange their models and aggregates via a federated averaging at a power consumption of 100 W.
3) *Model training*—If neither of the above takes place, the satellites perform a training epoch at a power consumption of 100 W.

2) Learned Model and Communications: Overall, training the models on the two satellites is successful, reaching an accuracy of over 91% on both satellites compared to a random guess accuracy of 50% and independent training that results in an accuracy below 70%. Fig. 10 displays the test set accuracy over time and shows how both satellites struggle to obtain good results individually. However, they noticeably improve their accuracy when they exchange models (marked with gray vertical lines). Notice that model transmission is virtually instant given the small models (only 1312 bits).

It can also be seen that each satellite benefits after each communication round as the test accuracy rapidly increases. However, once the satellite starts training on local data again, the performance deteriorates due to catastrophic forgetting [52]. Furthermore, there is a large discrepancy between the test accuracy for the two satellites. This happens as Satellite 1 is able to do training after each model exchange because charging is initiated after one exchange, and the satellite remains in LOS with the sun after the next exchange. Satellite 2, on the other hand, is charging before one exchange and has stopped charging during the next. Hence, Satellite 2 may not have a battery to train after one of the model exchanges and, therefore, its convergence is slower, as seen in Fig. 10.

In terms of power consumption, displayed in Fig. 11, we can observe training in the intervals of rapid oscillation of the SoC when the battery is charged and, consequently, the training run. During eclipse, the satellites go into standby to conserve SoC and drain only 2 W. Finally, the rapid increase in SoC occurs when a satellite faces the sun and training is not allowed because SoC is below 0.5.

D. Onboard Machine Learning for EO Using PASEOS

In addition to the experiments presented in this work, PASEOS has been utilized in two parallel studies that aim to train machine learning models on board satellite constellations, taking into account the constraints imposed by the space environment [53]. In our work, we trained an EfficientNet-lite0 [54] model on the EuroSAT [55], [56] dataset on board a constellation of 8 Sentinel-2-like satellites. We compared the training results

for three different decentralized training scenarios pertaining to i) federated learning using ground stations, ii) federated learning via a geostationary satellite, and iii) peer-to-peer federated learning.

In another study [53], we explored a new mission concept leveraging a deep compression neural network to provide highly compressed image previews of Sentinel-2 raw data [49]. These previews can be processed by end users on the ground, who can then request the download of the uncompressed images of interest. The model, a deep autoencoder provided by CompressAI [57], is trained on board a constellation of three Sentinel-2-like satellites and offers advantageous compression/quality tradeoffs compared to JPEG.

In both these parallel works, PASEOS was instrumental to monitor the impact of communication, power, and other constraints on the training loss and training time. Moreover, in comparison to the example in Section III-C, both of these studies adopted real EO datasets and deeper neural network models, further demonstrating the versatility of PASEOS to model the impact of space environment for scenarios using machine learning on board spacecraft for EO missions.

Moreover, in both these works PASEOS was used together with MPI to enable parallel and asynchronous simulations in which the different satellites are run as parallel nodes. This aspect further enhances the suitability of PASEOS to handle the simulation of multiple spacecraft.

IV. Discussion

In general, the examples in Section III clearly demonstrate the viability of PASEOS for the three considered test cases. By running on an actual satellite processor in real time, we have also shown the feasibility of using PASEOS to model scenarios while utilizing real, prospective mission hardware. In the case of the LEO constellation, we can see that PASEOS is also capable of modeling large constellations in a longer time frame. Finally, we also showcased how PASEOS can be used to model constraints that directly impact the training of machine learning models in space. There are, however, some aspects that mandate further discussion and consideration.

A. Impact of Onboard Constraints on Machine Learning

One of the objectives of PASEOS is to study the impact of operational constraints when utilizing machine learning methods in orbit. Especially in the context of distributed and onboard learning paradigms, it is an essential question what this impact will be [13], [14]. As demonstrated in our onboard learning results in Section III-C, PASEOS can provide concrete insights into the operational impact of factors such as orbital dynamics and power consumption. As can be seen in Fig. 10, the timing of communication windows and eclipse directly influenced the accuracy obtained during the distributed training. While the importance of factors such as temperature, battery SoC, or the timing of communication windows in relation to eclipse are well-established parameters for spacecraft operators, they have not been studied in the context of applying machine learning methods in orbit. Even in-orbit demonstrations, such as Φ-Sat-1 [8], did not explore the topic of continuous operations of these systems over a longer operational time frame where many of the constraints captured by PASEOS play an important role. For both inference and training in constellations, to the authors' knowledge, there is currently a lack of holistic modeling of these factors as offered by PASEOS. Our results already show the importance of this aspect and given its impact even in a relatively simple scenario with few devices (as witnessed in Fig. 10), the study of machine learning applications in orbit may benefit from using PASEOS.

B. Scalability and Performance

As shown on the single device example using real space-rated embedded hardware in Section III-A, PASEOS is a lightweight tool that allows modeling the space environment and operational constraints with minimal overhead for the user activity. Results shown in Section III-A3 showcase that the computational complexity inside PASEOS models scales linearly with the PASEOS timestep while requiring less than 1.5% of the total CPU time compared to the user activity with the highest investigated update rate of PASEOS. This is due to the particularly efficient physical models that offer suitable tradeoffs to be used on edge devices. This partially comes at the cost of model fidelity, for which the main limitations and possible future improvements are discussed in Sections IV-C and IV-D. Thus, PASEOS can enable studying applications and operational constraints directly on the target spacecraft hardware as demonstrated on the Unibap iX10-100 devices.

On the other end of the simulation spectrum, the LEO constellation example in Section III-B shows that even as PASEOS is able to handle large constellations of up to hundreds of satellites, performance does become a concern. If one wants to model the long-term viability of a constellation, including aspects like station keeping, degradation of photovoltaic cells and batteries, and similar effects (currently not implemented in PASEOS), computational time may become a concern. In the single-threaded, single-core run used for the example, the runtime would, at the moment, already be prohibitive for a constellation with hundreds of satellites on a timescale of months or years. PASEOS Version 0.1.3 fully supports MPI and is implemented in a fully asynchronous manner. Thus, it is straightforward to use it on compute clusters—this is shown in one of the online examples and the recent work [24]. A more detailed investigation of scaling on supercomputers is warranted.

In conclusion, PASEOS is meticulously crafted to facilitate both hardware-in-the-loop (HIL) simulations—as exemplified in Section III-A—and software-in-the-loop (SIL) simulation—as demonstrated in Section III-C. On a broader scale, PASEOS can be employed with arbitrary hardware via user-supplied Python interfaces, can be executed on actual space hardware, and can cosimulate in conjunction with external software, including machine learning applications.

C. Fidelity Considerations

One key point in terms of fidelity that must be considered are the user-provided specifications. They are not studied here

in detail as they are beyond the scope of this initial release. However, given the holistic nature and the complex emergent behavior—see, e.g., the complex SoC curve stemming from just a simple, linear charge model in Fig. 6—of PASEOS, the accuracy and quality of the input parameters is likely to become a critical factor in the fidelity of results produced by PASEOS. Notably, if all physical models are activated, the number of input parameters becomes fairly large. Indeed, just the thermal model requires eight parameters, the power model currently requires three, the radiation model another three, the orbital model seven, and these parameters need to be defined for every single actor in PASEOS. Thus, even with the simplified models present in PASEOS, it is evident that constellation scenarios modeled with PASEOS are already highly complex systems. In the future, this may require a more detailed analysis of the system's sensitivity to specific parameters to guide users as to which parameters are particularly critical. Currently, users can see an overview of all parameters, their likely sensitivity, and suggested ranges and default values in the documentation. It is also conceivable to add noise terms in a variety of the PASEOS models to account for this sensitivity. Even higher robustness code could be achieved by converting some of the parameters to be described by distributions instead of scalar parameters and thus include a probabilistic, Bayesian modeling component that allows consideration of prior assumptions about the accuracy of passed parameters.

D. Limitations and Prospective Additions

At the moment, there are some natural limitations to the fidelity of the models in PASEOS and supported scenarios. The two objectives of holistically modeling the operational environment in space and being able to execute PASEOS in the background in real time on edge devices requires a careful balancing of computational cost and physical fidelity. This may be remedied in the future by adding optional components to PASEOS that model the various physical aspects at a higher fidelity when enough computational resources are available. Initially, however, we have focused on the breadth of considered physical aspects instead of high fidelity in individual aspects. More expensive models can already be integrated via the *custom properties* described in Section II-A8.

In terms of astrodynamics modeling, Keplerian dynamics are a sufficient start for low-precision Earth orbits but insufficient to, e.g., model orbits around irregular bodies such as asteroids or comets. Similarly, phenomena such as station keeping and occasional losses of tracking cannot be modeled with them. Users should use the SGP4 propagator for higher fidelity by providing two-line elements. In addition, they can wrap models such as a polyhedral gravity model [58] or software like *orekit* [20] via the custom propagator option.

The availability of communication windows in PASEOS is currently determined solely by whether the LOS is being blocked by an assumed sphere with a specific radius. More complex geometry could be integrated via meshes. Other factors such as the success of tracking, the distance of the actors, and atmospheric conditions for optical communications and similar are also currently not modeled but sensible additions. Similarly, the communication bandwidth that is currently available is, in reality, a more complex and variable quantity dependent on a lot of the just mentioned factors, such as distance, link conditions, and other parameters linked to the transmission chain (i.e., channel-encoding, modulation, additional use of synchronization pilots, etc.). A more thorough channel modeling is required to account for this. A potential way forward is to wrap *ns-3* or *OMNeT++* into PASEOS [18], [19] via the *custom properties*.

In regards to power budgets, there are also several conceivable improvements, such as more rigorous modeling of the SoC of the battery [59], a more thorough model for the charging via solar panels [60] and consideration of factors such as the age and temperature of the battery, devices, and solar panels. In a similar vein, the thermal model in PASEOS would benefit from a more complex model that accounts for the thermal properties of various components such as solar panels, radiators and others. Concerning radiation, TID and total non-ionization dose (TNID) effects are not currently implemented in PASEOS, but they are relevant for simulations in the time scales of years [15], especially for commercial off-the-shelf devices [15], [28], [61]. However, we provide an example showing how one can model TID effects as *custom property* for a satellite moving along the Sentinel-2B orbit for a 30 days-long mission starting on 2022-10-27T12:00:00Z and estimate the consequent probability of a failure by using the approach described in [62] and [63]. We used SPENVIS [64] to estimate the expected TID for the mission. A similar approach can be used to model the TNID effects when needed.

Another direction of improvement lies in expanding the capabilities of actors. Ground-based actors are currently stationary and only supported for scenarios on Earth. Further, space-based actors cannot perform manoeuvres (although one can manually change the orbit). These capabilities could support more complex operational scenarios and activities in the future.

Finally, current examples using PASEOS mainly focus on use cases at the application level (e.g., detection of volcanic eruption, implementation of decentralized learning) or simulations of satellite constellations. At the moment PASEOS does not offer a model for spacecraft platform-level elements, such as attitude and orbit determination systems, or accurate models for sensors or actuators.

However, it should be noted that a model for these systems/devices can be implemented by leveraging the activity paradigm and *custom properties* features.

Overall, there is a virtually endless range of potential improvements in fidelity and modeled aspects. It will require careful analysis of which aspects are critical to enable realistic constraint consideration and which ones can be simplified to the degree currently in PASEOS. But, to allow these comparisons, one needs to start with a baseline, which we are providing here.

V. Conclusion

PASEOS is a software package that enables holistic modeling of the onboard environment accounting for, e.g., power, thermal, radiation, and dynamics. The generality of PASEOS makes it

a tool for studying many operational scenarios and hardware configurations or for use in conjunction with other simulation tools. In particular, PASEOS is well-suited to study constellations in space for emerging operational scenarios, such as edge computing, edge and decentralized learning, and artificial intelligence in space [65].

Overall, we have demonstrated that PASEOS provides the means to model a variety of constraints that spacecraft and their operators experience in orbit. Thus, we can explore the feasibility of onboard activities with greater rigor before launch and/or form an understanding of how already operational assets may be repurposed or performed in the future. Given PASEOS's asynchronous and versatile setup, a broad range of scenarios from one to multiple spacecraft (including ground-based actors) are possible. Both real-time at-the-edge execution and long-term simulation on a computing cluster or similar infrastructure are supported. The specifics of the modeled quantities can be adjusted to fit the particular scenario.

However, there are, of course, intrinsic limitations to this process. At the moment, the models inside PASEOS exhibit comparatively lower fidelity than simulators for dedicated topics (e.g., *ns-3* or *OMNeT++*) to enable rapid background execution. PASEOS's modular nature does provide a natural surface for the extension and wrapping of more complex physical models as enabled via *custom properties*, user-defined constraints and activities, as well as a custom propagator. In the future, we will conduct application-specific studies, explore more complex models, and demonstrate an operational scenario using PASEOS on multiple edge devices in parallel to solve a real-world task.

Acknowledgment

The authors would like to thank Unibap AB for providing the iX-10 100 device used for our experiments. Further, the authors would like to thank Bho Matthiesen and his team for the inspiring and insightful discussions.

References

[1] S. Lemmens and F. Letizia, "ESA's annual space environment report," Technical report, Tech. Rep. GEN-DB-LOG-00288-OPS-SD, ESA Space Debris Office, 2022.
[2] H. R. Goldberg et al., "The Juventas CubeSat in support of ESA's Hera mission to the asteroid didymos," 2019.
[3] J. D. Liddle, A. P. Holt, S. J. Jason, K. A. O'Donnell, and E. J. Stevens, "Space science with cubesats and nanosatellites," *Nature Astron.*, vol. 4, no. 11, pp. 1026–1030, 2020.
[4] H. Jones, "The recent large reduction in space launch cost," in *Proc. 48th Int. Conf. Environ. Syst.*, 2018.
[5] O. Kodheli et al., "Satellite communications in the new space era: A survey and future challenges," *IEEE Commun. Surv. Tut.*, vol. 23, no. 1, pp. 70–109, Firstquarter 2020.
[6] F. C. Bruhn, N. Tsog, F. Kunkel, O. Flordal, and I. Troxel, "Enabling radiation tolerant heterogeneous GPU-based onboard data processing in space," *CEAS Space J.*, vol. 12, no. 4, pp. 551–564, 2020.
[7] G. Giorgi et al., "Advanced technologies for satellite navigation and geodesy," *Adv. Space Res.*, vol. 64, no. 6, pp. 1256–1273, 2019.
[8] G. Giuffrida et al., "Cloudscout: A deep neural network for on-board cloud detection on hyperspectral images," *Remote Sens.*, vol. 12, no. 14, 2020, Art. no. 2205.
[9] M. P. Del Rosso et al., "On-board volcanic eruption detection through CNNs and satellite multispectral imagery," *Remote Sens.*, vol. 13, no. 17, 2021, Art. no. 3479.
[10] G. Mateo-Garcia et al., "Towards global flood mapping onboard low cost satellites with machine learning," *Sci. Rep.*, vol. 11, no. 1, Mar. 2021, Art. no. 7249.
[11] P. Gómez and G. Meoni, "MSMatch: Semisupervised multispectral scene classification with few labels," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 11643–11654, 2021.
[12] D. Izzo and P. Gómez, "Geodesy of irregular small bodies via neural density fields," *Commun. Eng.*, vol. 1, no. 1, pp. 1–12, 2022.
[13] N. Razmi, B. Matthiesen, A. Dekorsy, and P. Popovski, "On-board federated learning for dense LEO constellations," in *Proc. IEEE Int. Conf. Commun.*2022, pp. 4715–4720.
[14] B. Matthiesen, N. Razmi, I. Leyva-Mayorga, A. Dekorsy, and P. Popovski, "Federated learning in satellite constellations," *IEEE Netw.*, vol. 38, no. 2, pp. 232–239, Mar. 2024.
[15] G. Furano et al., "Towards the use of artificial intelligence on the edge in space systems: Challenges and opportunities," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 35, no. 12, pp. 44–56, Dec. 2020.
[16] J. R. Wertz, W. J. Larson, D. Kirkpatrick, and D. Klungle, *Space Mission Analysis and Design*, vol. 8. Berlin, Germany: Springer, 1999.
[17] G. Curzi, D. Modenini, and P. Tortora, "Large constellations of small satellites: A. survey of near future challenges and missions," *Aerospace*, vol. 7, no. 9, 2020, Art. no. 133.
[18] A. Varga and R. Hornig, "An overview of the omnet simulation environment," in *Proc. 1st Int. ICST Conf. Simul. Tools Techn. Commun., Netw. Syst.*, 2010, Art. no. 60.
[19] G. F. Riley and T. R. Henderson, "The NS-3 network simulator," in *Modeling and Tools for Network Simulation*. Berlin, Germany: Springer, 2010, pp. 15–34.
[20] L. Maisonobe, V. Pommier, and P. Parraud, "OREKIT: An open source library for operational flight dynamics applications," in *Proc. 4th Int. Conf. Astrodynamics Tools Techn.*, 2010, pp. 3–6.
[21] D. W. Walker and J. J. Dongarra, "MPI: A standard message passing interface," *Supercomputer*, vol. 12, pp. 56–68, 1996.
[22] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, vol. 32, pp. 8024–8035.
[23] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Conf. Operating Syst. Des. Implementation*, USENIX Association, USA, 2016, pp. 265–283.
[24] J. Östman, P. Gomez, V. M. Shreenath, and G. Meoni, "Decentralised semi-supervised onboard learning for scene classification in low-earth orbit," in *Proc. 14th IAA Symp. Small Satellites Earth Observ.*, 2023.
[25] A. Wall, "Systems tool kit (STK)," *Space Syst. Architecture Anal. Wargaming*, pp. 11–32, CRC Press.
[26] S. Ikari et al., "S2E: Spacecraft simulation environment," Software available from University of Tokyo, 2022. [Online] Available: https://github.com/ut-issl/s2e-documents
[27] P. M. Fisher et al., "Virtual satellite," Software available from German Aerospace Center (DLR), 2019. [Online] Available: https://github.com/virtualsatellite/VirtualSatellite4-Core
[28] G. Lentaris et al., "High-performance embedded computing in space: Evaluation of platforms for vision-based navigation," *J. Aerosp. Inf. Syst.*, vol. 15, no. 4, pp. 178–192, 2018.
[29] P. Fortescue, G. Swinerd, and J. Stark, *Spacecraft Systems Engineering*. Hoboken, NJ, USA: Wiley, 2011.
[30] D. Izzo, "PyGMO and PyKEP: Open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization)," in *Proc. 5th Int. Conf. Astrodynamics Tools Techn.*, 2012.
[31] D. Vallado and P. Crawford, "SGP4 orbit determination," in *Proc. AIAA/AAS Astrodynamics Specialist Conf. Exhib.*, 2008, Art. no. 6770.
[32] S. A. Al-Gailani et al., "Survey of free space optics (FSO) communication systems, links, and networks," *IEEE Access*, vol. 9, pp. 7353–7373, 2021.
[33] B. Rhodes, "Skyfield: Generate high precision research-grade positions for stars planets moons and earth satellites," 2020.
[34] H. Kaushal and G. Kaddoum, "Optical communication in space: Challenges and mitigation techniques," *IEEE Commun. Surveys Tut.*, vol. 19, no. 1, pp. 57–96, First Quarter 2017.
[35] C. S. Carrano, C. T. Bridgwood, and K. M. Groves, "Impacts of the december 2006 solar radio bursts on the performance of GPS," *Radio Sci.*, vol. 44, no. 01, pp. 1–12, 2009.
[36] G. Meoni et al., "CCSDS 131.2-B-1 telemetry transmitter: A. VHDL IP core and a validation architecture on board RTG4 FPGA," *Acta Astronautica*, vol. 176, pp. 484–493, 2020.
[37] G. Pitsis et al., "Efficient convolutional neural network weight compression for space data classification on multi-FPGA platforms," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2019, pp. 3917–3921.

[38] M. R. Patel, *Spacecraft Power Systems*. Boca Raton, FL, USA: CRC Press, 2004.
[39] I. Martinez, "Spacecraft thermal modelling and testing," vol. 43, 1995. [Online]. Available: http://webserver.dmt.upm.es/~isidoro/tc3/SpacecraftThermalModellingandTesting.pdf
[40] J. R. Srour and J. M. McGarrity, "Radiation effects on microelectronics in space," *Proc. IEEE IRE*, vol. 76, no. 11, pp. 1443–1469, Nov. 1988.
[41] Yifan Lu, Qi Shao, Honghao Yue, and Fei Yang, "A review of the space environment effects on spacecraft in different orbits," *IEEE Access*, vol. 7, pp. 93473–93488, 2019.
[42] O. Grasset et al., "Jupiter ICy moons explorer (JUICE): An ESA mission to orbit ganymede and to characterise the Jupiter system," *Planet. Space Sci.*, vol. 78, pp. 1–21, 2013.
[43] O. Özgün and Y. Barlas, "Discrete vs. continuous simulation: When does it matter," in *Proc. 27th Int. Conf. Syst. Dyn. Soc.*, 2009, vol. 6, pp. 1–22.
[44] M. Nugnes, C. Colombo, and M. Tipaldi, "Coverage area determination for conical fields of view considering an oblate earth," *J. Guid., Control, Dyn.*, vol. 42, no. 10, pp. 2233–2245, 2019.
[45] A. Arenas, A. Díaz-Guilera, J. Kurths, Y. Moreno, and C. Zhou, "Synchronization in complex networks," *Phys. Rep.*, vol. 469, no. 3, pp. 93–153, 2008.
[46] D. L. Mills, "Internet time synchronization: The network time protocol," *IEEE Trans. Commun.*, vol. 39, no. 10, pp. 1482–1493, Oct. 1991.
[47] O. Babaoglu, T. Binci, M. Jelasity, and A. Montresor, "Firefly-inspired heartbeat synchronization in overlay networks," in *Proc. IEEE 1st Int. Conf. Self-Adaptive Self-Organizing Syst.*, 2007, pp. 77–86.
[48] P. Di Stasio, A. Sebastianelli, G. Meoni, and Silvia Liberata Ullo, "Early detection of volcanic eruption through artificial intelligence on board," in *Proc. IEEE Int. Conf. Metrol. Extended Reality, Artif. Intell. Neural Eng.*, 2022, pp. 714–718.
[49] G. Meoni et al., "Unlocking the use of raw multispectral earth observation imagery for onboard artificial intelligence," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 17, pp. 12521–12537, 2024.
[50] F. Massimetti, D. Coppola, M. Laiolo, S. Valade, C. Cigolini, and M. Ripepe, "Volcanic hot-spot detection using sentinel-2: A comparison with Modis–Mirova thermal data series," *Remote Sens.*, vol. 12, no. 5, 2020, Art. no. 820.
[51] J. G. Walker, "Satellite constellations," *J. Brit. Interplanetary Soc.*, vol. 37, 1984, Art. no. 559.
[52] N. Shoham et al., "Overcoming forgetting in federated learning on non-IID data," 2019, *arXiv:1910.07796*.
[53] P. Gómez and G. Meoni, "Tackling the satellite downlink bottleneck with federated onboard learning of image compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 6809–6818.
[54] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.
[55] P. Helber, B. Bischke, A. Dengel, and D. Borth, "Introducing EuroSAT: A novel dataset and deep learning benchmark for land use and land cover classification," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2018, pp. 204–207.
[56] P. Helber, B. Bischke, A. Dengel, and D. Borth, "EuroSAT: A. novel dataset and deep learning benchmark for land use and land cover classification," *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.*, vol. 12, no. 7, pp. 2217–2226, Jul. 2019.
[57] J. Bégaint, F. Racapé, S. Feltman, and A. Pushparaja, "CompressAI: A pytorch library and evaluation platform for end-to-end compression research," 2020, *arXiv:2011.03029*.
[58] D. Tsoulis, "Analytical computation of the full gravity tensor of a homogeneous arbitrarily shaped polyhedral source using line integrals," *Geophysics*, vol. 77, no. 2, pp. F1–F11, 2012.
[59] X. Lai, S. Wang, S. Ma, J. Xie, and Y. Zheng, "Parameter sensitivity analysis and simplification of equivalent circuit model for the state of charge of lithium-ion batteries," *Electrochimica Acta*, vol. 330, 2020, Art. no. 135239.
[60] A. Porras-Hermoso, B. Cobo-Lopez, J. Cubas, and S. Pindado, "Simple solar panels/battery modeling for spacecraft power distribution systems," *Acta Astronautica*, vol. 179, pp. 345–358, 2021.
[61] W. S. Slater, N. P. Tiwari, T. M. Lovelly, and J. K. Mee, "Total ionizing dose radiation testing of Nvidia Jetson Nano GPUs," in *Proc. IEEE High Perform. Extreme Comput. Conf.*, 2020, pp. 1–3.
[62] M. A. Xapsos et al., "Inclusion of radiation environment variability in total dose hardness assurance methodology," *IEEE Trans. Nucl. Sci.*, vol. 64, no. 1, pp. 325–331, Jan. 2017.
[63] J.-z. Wang, Z.-xi Huo, and F.-f. Wang, "TID evaluation based on variabilities of space radiation and device failure dose in typical navigation satellite orbits," *Microelectron. Rel.*, vol. 137, 2022, Art. no. 114747.
[64] M. Kruglanski et al., "Space environment information system (SPENVIS)," in *Proc. EGU Gen. Assem. Conf. Abstr.*, 2009, Art. no. 7457.
[65] J. M. G. Sanchez, N. Jorgensen, and M. Torngren, "Edge computing for cyber-physical systems," *ACM Trans. Cybern.-Phys. Syst.*, vol. 6, no. 3, 2022, Art. no. 26.

Pablo Gómez received the M.Sc. degree in computer science from the Technical University Munich, Munich, Germany, in 2015 and the Ph.D. degree in electrical engineering in the topic deep learning methods for high-speed videoendoscopy and numerical modeling of the voice from the Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany, in 2019 (supervisor Prof. Döllinger).

He is a Data Scientist with ESA and previous research fellow in ESA's Advanced Concepts Team with a secondment to the national institute for artificial intelligence AI Sweden. His research topics of interest to him range from machine learning and inverse problems to numerical methods and high-performance computing.

Johan Östman received the M.Sc. and Ph.D. degrees in electrical engineering from the Chalmers University of Technology, Gothenburg, Sweden, in 2014 and 2020, respectively.

He is a Research Scientist with AI Sweden, Sweden. In 2015, he was a Research Scientist with Hong Kong University of Technology, Hong Kong. His main area of interest resides in the intersection of information theory, wireless communications, and decentralized learning.

Vinutha Magal Shreenath received the Ph.D. degree in AI, systems, design from KTH Royal Institute of Technology, Stockholm, Sweden, in 2019 and is an alumnus of Frontier Development Lab Europe.

She is the Chief Research Officer of Women in AI. Apart from AI in space, she works on design science and systems engineering. Prior to her Ph.D., she worked in the software industry for several years. She conducted the work presented in this article while she was at AI Sweden.

Gabriele Meoni (Member, IEEE) received the M.Sc. degree in electronic engineering and the Ph.D. degree in information engineering from the University of Pisa, Pisa, Italy, in 2016 and 2020, respectively, where he was supervised by Prof. Luca Fanucci.

After completing his doctoral studies, from 2020 to 2023 he held the position of Internal Research Fellow at the European Space Agency (ESA) (Advanced Concepts Team (ACT) 2020 to 2021, Φ-lab 2021 to 2023), where he conducted research on Artificial Intelligence (AI) and neuromorphic computing for onboard spacecraft applications. During 2022–2023, he was a Visiting Researcher at AI Sweden, focusing on distributed edge learning for satellite constellations. From 2023 to 2024, he was an Assistant Professor with the Faculty of Aerospace Engineering, Delft University of Technology. Currently, he is an Innovation Officer at ESA, with research interests spanning satellite onboard processing, AI for Earth Observation, and neuromorphic computing. He coauthored more than 40 scientific publications.