75th International Astronautical Congress (IAC), Milan, Italy, 14-18 October 2024.


**IAC-24,B6,3,8,x85387**

# PROX-SIMA: A modular simulator for the validation of In-Orbit servicing and close proximity missions GNC techniques

**Niccolò Faraco[a,*], Mauro Massari[c], Pierluigi Di Lizia[d]**

[a,*] *PhD Student, Politecnico di Milano, 20156 Milan, Italy,* niccolo.faraco@polimi.it
[b] Associate Professor, Politecnico di Milano, 20156 Milan, Italy, mauro.massari@polimi.it
[c] Associate Professor, Politecnico di Milano, 20156 Milan, Italy, pierluigi.dilizia@polimi.it

*Corresponding author

## Abstract

Among the ever-growing number of objects polluting the space around the Earth, the number of inactive satellites that could be put back into service after refuelling and/or minor repairing is significant. Therefore, from a reusability and sustainability standpoint, In-Orbit Servicing missions will play a substantial role in the upcoming decades. Designing such missions is particularly challenging from a GNC perspective. When the measurements for the navigation subsystem come from optical sensors, in particular, it is paramount to simulate data representative of what the sensors would acquire during flight, as to build algorithms capable of producing relevant and robust results. To ease the research in this field and the further development of GNC algorithms for proximity operations, we developed the PROXimity Spacecraft IMaging and Autonomous Systems SIMulation tool (PROX-SIMA).
The software relies on three main components. The mathematical model of the servicing system is described using a Functional Mockup Unit (FMU), which is provided by the user. This allows for a thorough description of the system and its relative motion with respect to the target, while still retaining computational efficiency during simulation. Moreover, relying on the FMU standardized interface and the broad range of dedicated modeling softwares, enhances flexibility on the user side. The generation of the images from the sensors is instead entrusted to the Unreal Engine game engine, known for its photorealistic real-time rendering capabilities. Lastly, a Python server handles the communication between the FMU and the visualization tool, transmitting to the game engine the pose of the sensors as obtained from the FMU and getting back the images for processing. Such a plug-n-play framework allows for the user to focus on the development of the image processing, path-planning and closed-loop control algorithms, avoiding the need to set up ad-hoc simulations and enabling a more efficient, fast, and systematic exploration of different solutions.
In the first prototype of the pipeline, the model for a space manipulator system is implemented as an example and the generated images account for the presence of two monocular cameras mounted, respectively, on the base of the chasing spacecraft and on the end-effector of the manipulator. Nonetheless, the architecture of the software is geared towards the subsequent implementation of other models both for the space system (e.g. different satellites platforms) and different sensors (e.g. infrared cameras), as well as supporting hardware-in-the-loop simulations.
PROX-SIMA is planned to be made publicly available in its beta version.

**Keywords**: Computer Vision, In-orbit servicing, Modeling and simulation, Stereo Vision, Unreal Engine.

## 1. Introduction

The increasing congestion of Earth's orbital environment with space debris and inactive satellites has become a pressing concern for the global space community [1]. Among the multitude of objects orbiting our planet, a significant number are inactive satellites that could potentially be refurbished and returned to service through refueling or minor repairs [2]. This realization has sparked growing interest in In-Orbit Servicing (IOS) missions, which are poised to play a crucial role in space sustainability efforts over the coming decades [3]. In-Orbit Servicing encompasses a wide range of operations, including inspection, refueling, repair, upgrade, and debris removal [4]. These missions offer numerous benefits, such as extending the operational lifespan of satellites, reducing space debris, and potentially decreasing the overall cost of space operations [5]. However, the design and execution of IOS missions present unique challenges, particularly in the domain of Guidance, Navigation, and Control (GNC) [6]. One of the most critical aspects of IOS missions is the proximity operations phase, during which the servicing spacecraft must approach and potentially dock with the target object [7]. This phase requires extremely precise relative navigation [8, 9] and control [10, 11] algorithms, often relying on data from optical sensors to determine the relative position and ori-

entation of the two spacecraft [12, 13, 14]. The development of robust GNC algorithms for these operations necessitates high-fidelity simulations that can accurately represent the complex dynamics of orbital rendezvous and the visual environment encountered by optical sensors in space [15]. Existing simulation tools for space operations often fall short in providing a comprehensive environment for testing and validating GNC algorithms for proximity operations. Many focus either on high-fidelity dynamics simulations or on generating realistic visual data, but rarely combine both aspects effectively [16, 17, 18]. Additionally, the integration of custom spacecraft models and sensor configurations can be cumbersome, limiting the flexibility required for exploring diverse mission scenarios [19]. To address these challenges and facilitate research in the field of IOS, we have developed the PROXimity Spacecraft IMaging and Autonomous Systems SIMulation tool (PROX-SIMA). This innovative software platform combines high-fidelity dynamic modeling with photorealistic real-time rendering capabilities, providing researchers and engineers with a powerful and flexible environment for developing and testing GNC algorithms for proximity operations. PROX-SIMA builds upon the concept of modular simulation frameworks, similar to the approach described by Tasora and Mangoni [20] for vehicle simulations. However, our tool is specifically tailored for the unique challenges of space operations.

In the following sections, we will describe the architecture and implementation of PROX-SIMA in detail, including the modeling approach for the space manipulator system used as an initial test case. We will also discuss the current capabilities of the software, its potential applications in IOS mission design and GNC algorithm development, and outline plans for future enhancements and public release of the tool. By providing a comprehensive and flexible simulation environment for proximity operations in space, PROX-SIMA aims to accelerate research and development in the field of In-Orbit Servicing. As the space industry continues to evolve towards more sustainable practices, tools like PROX-SIMA will play a crucial role in enabling the next generation of space missions and technologies.

## 2. Software architecture

The software architecture of PROX-SIMA is based on three main components:

1. A Functional Mockup Unit (FMU) that encapsulates the mathematical model of the servicing system, allowing for detailed description of the spacecraft dynamics while maintaining computational efficiency.

2. The Unreal Engine game engine, which provides state-of-the-art photorealistic rendering capabilities for generating realistic sensor images in real-time.

3. A Python server that manages communication between the FMU and the visualization tool, ensuring seamless integration of the dynamics simulation with the visual environment.

This modular architecture, graphically illustrated in Fig. 1, offers several advantages. First, the use of FMUs, which adhere to the Functional Mock-up Interface (FMI) standard [21], enables users to leverage a wide range of modeling tools and libraries to create custom spacecraft models. This flexibility allows for the exploration of various mission scenarios and spacecraft configurations without the need to modify the core simulation framework. Second, the employment of Unreal Engine for image generation capitalizes on the rapid advancements in real-time rendering technology driven by the gaming industry [22]. This approach enables the simulation of complex lighting conditions, surface materials, and camera effects that are crucial for developing robust vision-based navigation algorithms. Lastly, the Python server acts as a bridge between the dynamics simulation and the visualization engine, facilitating data exchange and synchronization. This decoupled architecture allows for future expansions, such as the integration of hardware-in-the-loop simulations or the incorporation of additional sensor types.

### *2.1 Modelica and Functional Mockup Units*

The choice of modeling language and simulation framework is crucial for the accurate representation of complex dynamic systems such as spacecraft. In PROX-SIMA, we leverage the power of Modelica and Functional Mockup Units (FMUs) to provide a flexible and efficient simulation environment. Modelica is an object-oriented, equation-based modeling language specifically designed for multi-domain physical systems [23]. It offers several advantages for space systems modeling:

- Acausal modeling: Modelica allows for acausal (non-directed) connections between components, which naturally represent physical connections in spacecraft systems, such as mechanical joints or electrical circuits.

- Component reusability: The object-oriented nature of Modelica facilitates the creation of reusable component libraries, enabling rapid prototyping and testing of different spacecraft configurations [24].

- Multi-domain integration: Modelica seamlessly integrates models from different physical domains (mechanical, electrical, thermal, etc.), which is essential for comprehensive spacecraft modeling [25].

- Automatic equation handling: Modelica tools automatically manipulate and optimize the system equa-

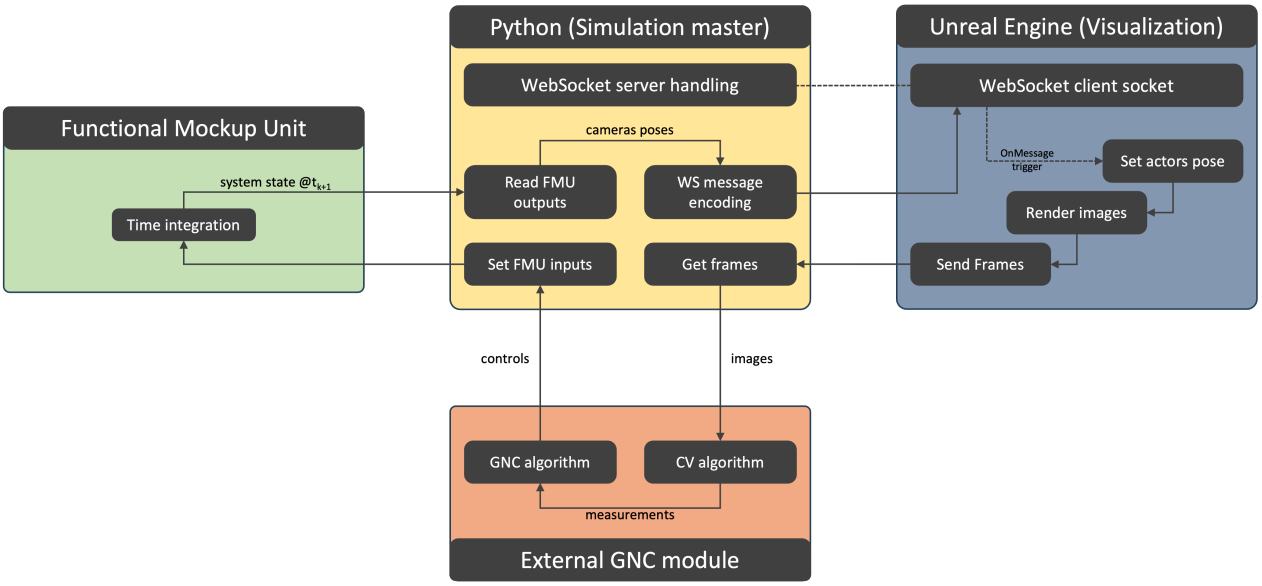


**Fig. 1:** PROX-SIMA architecture.

tions, reducing the likelihood of errors in complex models and improving simulation efficiency [26].

The FMI standard defines a standardized interface for model exchange and co-simulation [21]. FMUs, which are compiled versions of models that conform to the FMI standard, offer several benefits in the context of spacecraft simulation:

- Tool independence: FMUs can be generated by various modeling tools and used in different simulation environments, promoting interoperability and flexibility in the development process [27].
- Intellectual property protection: The compiled nature of FMUs allows for the sharing of models without exposing proprietary details, facilitating collaboration in multi-organization projects.
- Efficient execution: FMUs can be optimized for numerical efficiency, crucial for real-time simulations of complex spacecraft dynamics [28].
- Easy integration: The standardized FMI API simplifies the integration of models into larger simulation frameworks, such as PROX-SIMA [29].

For our project, the combination of Modelica and FMUs provides an ideal foundation. The expressive power of Modelica allows for detailed modeling of spacecraft systems, including rigid body dynamics, attitude control, and various subsystems. The resulting models can then be compiled into FMUs, which our Python server can efficiently execute and integrate with the visual simulation in Unreal Engine. This approach strikes a balance between model fidelity, computational efficiency, and integration flexibility, making it well-suited for the demands of proximity operations simulation in In-Orbit Servicing scenarios.

### 2.2 *Unreal Engine for visual simulation*

In the presented work, we use Unreal Engine to provide high-fidelity visual simulation capabilities. Unreal Engine is a powerful real-time 3D creation tool originally developed for video games but increasingly used in various industries, including aerospace and robotics simulations and scientific simulations of various nature. It offers several key advantages for space systems visualization:

- Photo-realistic rendering: Unreal Engine's advanced rendering capabilities, including real-time ray tracing and physically-based materials, enable the creation of highly realistic space environments and spacecraft models [30].
- Real-time performance: The engine's optimized rendering pipeline allows for smooth, real-time visualization of complex scenes, crucial for interactive simulations and algorithm testing [22].
- Extensive asset library: Unreal Engine provides a vast marketplace of pre-made assets and tools, accelerating the development of visual simulations [31].
- Programmable pipeline: The engine's Blueprint visual scripting system and C++ API allow for custom shader development and post-processing effects, enabling accurate simulation of various camera and sensor types.

- VR/AR support: Built-in virtual and augmented reality capabilities facilitate immersive visualization and potential integration with hardware-in-the-loop setups [32].

Compared to other 3D creation tools like Blender, Unreal Engine's real-time performance and specialized features for interactive applications make it particularly well-suited for the dynamic and responsive nature of spacecraft proximity operations simulations. Its robust development ecosystem and continuous updates ensure that PROX-SIMA can leverage cutting-edge visualization technologies as they become available.

### *2.3 Implementation details*

An high-level overview of the pipeline is presented in Fig. 1.
The software architecture follows a microservices-inspired approach, wherein each component is assigned a specific, well-defined task, and inter-component communication is facilitated through an API structure. The Python code serves as the simulation orchestrator, initializing a WebSocket server[1] at runtime to manage communication between various components.

It is crucial to reiterate that PROX-SIMA is conceived as a tool to facilitate research in Guidance, Navigation, and Control (GNC) for proximity missions. The GNC module depicted in the figure represents the researcher's or user's area of study and is not an inherent component of the presented tool.
As the external GNC module generates control inputs for the subsequent time step, these are transmitted to the Functional Mock-up Unit (FMU). The FMU then performs time integration and computes the system state evolution. Subsequently, the simulation master extracts the relevant data from the FMU's state, encodes it into a WebSocket message, and relays it to Unreal Engine, which is responsible for visualization.

On the visualization module side, upon game launch, a WebSocket client establishes a connection to the server and continuously monitors incoming messages. When a message arrives containing information about the scene's camera poses, it is decoded, and the position and orientation of the relevant actors (chaser, target, and cameras) are updated accordingly. The images from each camera are then rendered, appropriately encoded, and transmitted back to the Python server via the WebSocket connection for them to be forwarded to the GNC module.

Given that both the GNC module and the FMU are user-provided, interface functions are supplied to enable researchers to define the logic of their GNC algorithm and specify the FMU variables pertinent to the computation.

[1] https://websockets.spec.whatwg.org/ [last accessed on September, 17th, 2024]

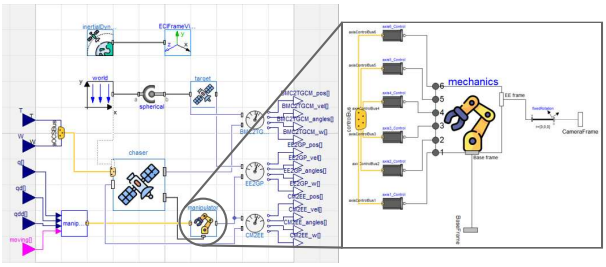


**Fig. 2:** Full Modelica model of the chasing spacecraft system and dynamical environment (left) and detail of the manipulator model (right).

In the current implementation, users must manually load the target satellite model into the Unreal Engine level. Future software updates may automate this process, requiring users to only specify the filepath of the desired model.

## 3. Example application

The system on which the pipeline is tested is constituted by a so-called Space Manipulator System (SMS), i.e. a satellite endowed with a robotic manipulator mounted on board. The system is assumed to be equipped with two monocular cameras, one mounted on the base of the satellite and the other on the end-effector of the manipulator. PROX-SIMA is entitled to simulate the images that the two cameras would acquire in orbit and make them available to the researcher/user of the software so it can be used as a source of measurements for any GNC implementation currently under development. The GNC algorithm and development fall outside the scope of the present paper.

### *3.1 FMU model*

The high-fidelity simulation FMU is built using the Modelica language[2] and the Dymola modeling tool[3].
Figure 2 illustrates the comprehensive model structure employed in our simulation.

The Modelica framework and its Standard Library facilitate the modeling of our complex space manipulator system by integrating various subsystems, including joint control assemblies with their associated sensors and actuators. While the intricate details of each manipulator component's mechanical implementation are beyond the scope of this paper, it's worth noting that the standard Modelica library has certain limitations. Specifically, the acceleration parameters for the default Body objects in Modelica are restricted to the "world" model frame (i.e., the geocentric inertial frame). To address this constraint,

[2] https://modelica.org/ [last accessed on September, 17th, 2024]

[3] https://www.3ds.com/products/catia/dymola [last accessed on September, 17th, 2024]

we developed a specialized subclass of the standard body object. This custom class incorporates a dynamic acceleration module capable of computing accelerations for relative motion scenarios between satellites, drawing upon the work of Franzini and Innocenti [33]. This approach allows us to more accurately model the complex dynamics of proximity operations in space, while maintaining the flexibility and modularity offered by the Modelica environment.

### *3.2 Simulation results*

A representative sequence of images generated by PROX-SIMA is shown in Fig. 3, exemplifying the pipeline's output. While the study of GNC algorithms falls beyond the scope of this work, for demonstration purposes, the GNC module's output is simplified to a single impulse. This impulse initiates the chaser's motion, along with its cameras, in an orbit around the target spacecraft. For conciseness and clarity, only the images from the body-mounted camera are presented here.

## 4. Conclusions

PROX-SIMA represents a significant advancement in simulation tools for In-Orbit Servicing missions. By integrating high-fidelity dynamic modeling via Functional Mock-up Units with Unreal Engine's photo-realistic rendering, it offers a powerful platform for developing and testing GNC algorithms for proximity operations.

The software's modular architecture, centered around a Python server, provides flexibility for integrating custom spacecraft models and sensor configurations. The initial implementation, featuring a Space Manipulator System with two monocular cameras, demonstrates the tool's potential for complex IOS scenario simulation.

It is crucial to note that PROX-SIMA is still under heavy development. While core functionality has been implemented and tested, several areas for future development exist, including:

- Automating the integration of target satellite models into Unreal Engine.
- Expanding the sensor suite to include LiDAR and infrared cameras.
- Implementing more sophisticated space lighting models.
- Implementing hardware-in-the-loop capabilities.

As PROX-SIMA evolves, we anticipate it becoming a valuable asset for the space community, accelerating the development of robust GNC solutions for future In-Orbit Servicing missions. We welcome collaboration and feedback as we work towards a public release of the software.

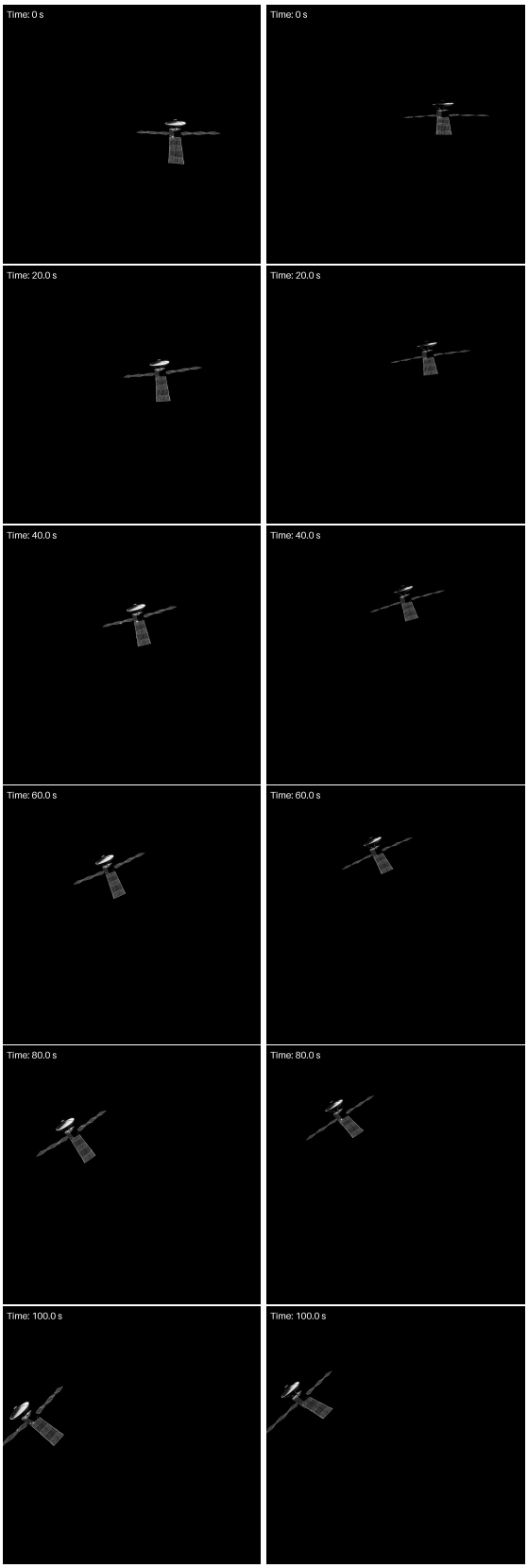


**Fig. 3:** Sequences of captured images from the body-mounted camera (left column) and end-effector camera (right column) during proximity motion.

## References


[1] Kessler DJ, Cour-Palais BG. Collision frequency of artificial satellites: The creation of a debris belt. *Journal of Geophysical Research: Space Physics*, 1978, 83(A6): 2637–2646, doi:https://doi.org/10.1029/JA083iA06p02637.

[2] Liou JC, Johnson NL. Risks in Space from Orbiting Debris. *Science*, 2006, 311(5759): 340–341, doi:10.1126/science.1121337.

[3] Flores-Abad A, Ma O, Pham K, Ulrich S. A review of space robotics technologies for on-orbit servicing. *Progress in aerospace sciences*, 2014, 68: 1–26.

[4] Ellery A, Kreisel J, Sommer B. The case for robotic on-orbit servicing of spacecraft: Spacecraft reliability is a myth. *Acta Astronautica*, 2008, 63(5): 632–648, doi:https://doi.org/10.1016/j.actaastro.2008.01.042.

[5] Saleh JH, Lamassoure E, Hastings DE. Space Systems Flexibility Provided by On-Orbit Servicing: Part 1. *Journal of Spacecraft and Rockets*, 2002, 39(4): 551–560, doi:10.2514/2.3844.

[6] Fehse W, 2003. *Automated Rendezvous and Docking of Spacecraft*. Cambridge Aerospace Series. Cambridge University Press.

[7] Bonnal C, Ruault JM, Desjean MC. Active debris removal: Recent progress and current trends. *Acta Astronautica*, 2013, 85: 51–60, doi:https://doi.org/10.1016/j.actaastro.2012.11.009.

[8] Maestrini M, De Luca MA, Di Lizia P. Relative navigation strategy about unknown and uncooperative targets. *Journal of Guidance, Control, and Dynamics*, 2023, 46(9): 1708–1725, doi:10.2514/1.G007337.

[9] Maestrini M, Di Lizia P. COMBINA: Relative Navigation for Unknown Uncooperative Resident Space Object. In *AIAA Scitech Forum, 3-7 January 2022, San Diego, CA*, AIAA, 1–19, doi:10.2514/6.2022-2384, Paper No. AIAA 2022-2384.

[10] Zhao C, Maestrini M, Di Lizia P. Low-Thrust Optimal Control of Spacecraft Hovering for Proximity Operations. *Journal of Guidance, Control, and Dynamics*, 2024, 47(7): 1457–1469, doi:10.2514/1.G008063.

[11] Maestrini M, Di Lizia P, Topputo F. Analytical Impulsive-to-Continuous Thrust Conversion in Linearized Relative Dynamics. *Journal of Guidance, Control, and Dynamics*, 2021, 44(4): 862–871, doi:10.2514/1.G005520.

[12] Opromolla R, Fasano G, Rufino G, Grassi M. A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations. *Progress in Aerospace Sciences*, 2017, 93: 53–72, doi:https://doi.org/10.1016/j.paerosci.2017.07.001.

[13] Razgus B, Maestrini M, Lizia PD. Initial Attitude Acquisition for Uncooperative Resident Space Objects Using Principal Lines. In *AIAA SCITECH Forum, 8-12 January 2024, Orlando, Florida*, AIAA, 1–8, doi:10.2514/6.2024-0201, Paper No. AIAA 2024-0201.

[14] Kaidanovic D, Piazza M, Maestrini M, Di Lizia P. Deep learning-based relative navigation about uncooperative space objects. In *Proceedings of the 73rd International Astronautical Congress, 18-22 September, 2022, Paris, France.*, volume 2022-September, IAF, Paper No. IAC-22.C1.IPB.38.x70501.

[15] Petit A, Marchand E, Kanani K. Vision-based space autonomous rendezvous: A case study. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, IEEE, Piscataway, New Jersey., 619–624.

[16] Pugliatti M, Maestrini M. A multi-scale labeled dataset for boulder segmentation and navigation on small bodies. In *Proceedings of the 74th International Astronautical Congress, 02-06 October, 2023, Baku, Azerbaijan.*, volume 2023-October, IAF, Paper No. IAC-23,A3,4A,8,x78927.

[17] Pugliatti M, Maestrini M. Small-Body Segmentation Based on Morphological Features with a U-Shaped Network Architecture. *Journal of Spacecraft and Rockets*, 2022, 59(6): 1821 – 1835, doi:10.2514/1.A35447.

[18] Faraco N, Maestrini M, Di Lizia P. Instance segmentation for feature recognition on noncooperative resident space objects. *Journal of Spacecraft and Rockets*, 2022, 59(6): 2160–2174, doi:10.2514/1.A35260.

[19] Rems F, Risse EA, Benninghoff H. Rendezvous GNC-System for Autonomous Orbital Servicing of Uncooperative Targets. In *10th International ESA Conference on Guidance, Navigation & Control*, 2017.

[20] Tasora A, Mangoni D. A Photorealistic Rendering Infrastructure for Man-in-the-Loop Real-Time Vehicle Simulation. In *Proceedings of SIMUL 2021, The Thirteenth International Conference on Advances in System Simulation*, 2021.

[21] Blochwitz T, Otter M, Arnold M, Bausch C, Clauß C, Elmqvist H, Junghanns A, Mauss J, Monteiro M, Neidhold T, Neumerkel D, Olsson H, Peetz JV, Wolf S. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. In *Proceedings of the 8th International Modelica Conference*, 2011, 105–114, doi:10.3384/ecp11063105.

[22] Pharr M, Jakob W, Humphreys G, 2016. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition.

[23] Fritzson P, Bunus P. Modelica - a general object-oriented language for continuous and discrete-event system modeling and simulation. In *Proceedings 35th Annual Simulation Symposium. SS 2002*, IEEE, Piscataway, New Jersey, 365–380, doi:10.1109/SIMSYM.2002.1000174.

[24] Elmqvist H, Mattsson S, Otter M. Modelica-a language for physical system modeling, visualization and interaction. In *Proceedings of the 1999 IEEE International Symposium on Computer Aided Control System Design (Cat. No.99TH8404)*, 1999, 630–639, doi:10.1109/CACSD.1999.808720.

[25] Tiller M, 2001. *Introduction to Physical Modeling with Modelica*. Springer Science & Business Media.

[26] Fritzson P, 2014. *Principles of Object Oriented Modeling and Simulation with Modelica 3.3*. John Wiley & Sons, Ltd.

[27] Andersson C, Åkesson J, Führer C. PyFMI: A Python Package for Simulation of Coupled Dynamic Models with the Functional Mock-up Interface. In *Technical Report in Mathematical Sciences, 2016(2)*, 2016.

[28] Brembeck J, Pfeiffer A, Fleps-Dezasse M, Otter M, Wernersson K, Elmqvist H. Nonlinear State Estimation with an Extended FMI 2.0 Co-Simulation Interface. In *In Proceedings of the 10th International Modelica Conference*, 2014, 53–62, doi:10.3384/ecp1409653.

[29] Galtier V, Vialle S, Dad C, Tavella JP, Lam-Yee-Mui JP, Plessis G. FMI-Based Distributed Multi-Simulation with DACCOSIM. In *Symposium on Theory of Modeling and Simulation - TMS'15*, 2015, 804–811.

[30] Seyb D, Jacobson A, Nowrouzezahrai D, Jarosz W. Non-linear sphere tracing for rendering deformed signed distance fields. *ACM Trans. Graph.*, 2019, 38(6), doi:10.1145/3355089.3356502.

[31] Eleftheria C, Stelios X. Overview and Comparative Analysis of Game Engines for Desktop and Mobile Devices. *International Journal of Serious Games*, 2017, 4(4), doi:10.17083/ijsg.v4i4.194.

[32] Zyda M. From visual simulation to virtual reality to games. *Computer*, 2005, 38(9): 25–32, doi:10.1109/MC.2005.297.

[33] Franzini G, Innocenti M. Relative Motion Dynamics with Arbitrary Perturbations in the Local-Vertical Local-Horizon Reference Frame. *The Journal of the Astronautical Sciences*, 2020, 67(1): 98–112, doi:10.1007/s40295-019-00185-0.