














# ASA: A Simulation Environment for Evaluating Military Operational Scenarios

Joao P. A. Dantas<sup>(✉)</sup>, Andre N. Costa, Vitor C. F. Gomes,  
Andre R. Kuroswiski, Felipe L. L. Medeiros, Diego Geraldo,  
Adrisson R. Samersla, Samara R. Silva, Andre F. M. Caetano,  
Yuri D. Ferreira, Alessandro O. Arantes, Davison S. Santos,  
and Marcia R. C. Aquino

Decision Support Systems Subdivision, Institute for Advanced Studies,  
Trevó Coronel Aviador Jose A. A. do Amarante, 1, Putim, São José dos Campos,  
12228-001 São Paulo, Brazil

{dantasjpad,negraoanc,vitorvcfg,kuroswiskiark,felipeflm,diegodg,  
samerslaars,samarasrs,andreafmc,yuriydf,alessandroaoa1,  
davisondss,marciamrca}@fab.mil.br

**Abstract.** The Aerospace Simulation Environment (*Ambiente de Simulação Aeroespacial* – ASA in Portuguese) is a custom-made object-oriented simulation framework developed mainly in C++ that enables the modeling and simulation of military operational scenarios to support the development of tactics and procedures in the aerospace context for the Brazilian Air Force. This work describes the ASA framework and its main features: a distributed architecture for coordinating multiple simulation machines, a modular structure that allows models to be loaded at runtime, a batch execution mode for simulating multiple scenarios with varied initial conditions, and an integrated data analysis platform for post-processing simulation results. In addition, we present a list of recent studies that have used ASA in applications related to decision support and autonomy in air combat scenarios.

**Keywords:** simulation environment · distributed simulation · data analysis · military · operational scenarios

## 1 Introduction

The Institute for Advanced Studies (IEAv), a research organization of the Brazilian Air Force (*Força Aérea Brasileira* — FAB, in Portuguese), has developed, since 2018, the Aerospace Simulation Environment (*Ambiente de Simulação Aeroespacial* — ASA, in Portuguese) to provide a computational solution that enables the modeling and simulation of operational scenarios. This solution allows users to define strategies, parameters, and command decisions to support the development of tactics, techniques, and procedures in the aerospace domain for defense purposes.

The characteristics of modern battlefield scenarios present significant challenges to the development of practical combat simulations [2]. These challenges call for more integrated and flexible solutions that can address both technical and organizational aspects [19].

To meet some of these demands, frameworks such as the Advanced Framework for Simulation, Integration, and Modeling (AFSIM) [1], Wukong [18], FLAMES [29], and VR-Forces [25] have been developed. However, many of these tools are commercial products or restricted to use within specific countries. In this context, the ASA environment was conceived as a national solution designed to support FAB's strategic planning, meet operational analysis needs, and promote the development and evaluation of emerging technologies for military research.

ASA is designed as a flexible and modular platform, capable of adapting to diverse user needs. This flexibility is essential given the wide range of requirements from its users. These needs could not be fully addressed by commercial off-the-shelf (COTS) simulation software. Rather than building an entirely new system, the development approach integrated openly available tools into a unified simulation environment that is flexible, accessible, and scalable.

The proposed solution uses the Mixed Reality Simulation Platform (MIXR) [20] as its simulation engine, an open-source software project designed to support the development of robust, scalable, virtual, constructive, stand-alone, and distributed simulation applications. ASA extends MIXR's capabilities by adding components that streamline tasks for both developers and analysts. A manager application was created to serve as an interface between multiple resources, working as a hub to execute, store, and analyze simulations across multiple machines. This application also supports the simultaneous creation of numerous simulations by varying initial conditions according to the analyst's needs. Furthermore, models and tools can be dynamically loaded at runtime to increase flexibility. All simulation data are stored in a dedicated database, expediting data collection and enabling more robust statistical analysis. Additionally, given the complexity of simulation outcomes and the varied technical background of ASA users, a dedicated data analysis platform was integrated into the system, not only for planning and visualization but also for post-processing the scenario data.

The main contribution of this work is the introduction of a new environment for modeling and simulation in the aerospace domain for military applications. It features: (i) a distributed architecture for managing multiple simulation machines; (ii) support for runtime model loading within a modular architecture, allowing new models to be easily integrated; (iii) a batch execution mode that enables the simulation of multiple scenarios with varied initial conditions; and (iv) an enhanced data analysis platform for post-processing military operational scenario data. Additionally, we present a list of recent studies that have used the ASA platform in applications related to decision support and autonomy in military operational scenarios.

The remainder of this paper is organized as follows. Section 2 presents the ASA architecture. Section 3 discusses studies that have used ASA in air combat analysis as application examples of the simulation framework. Finally, Sect. 4 provides concluding remarks on the current state of ASA and outlines potential directions for future work.

## 2 ASA Architecture

The ASA design consists of three main modules. The first part is the simulation framework, defined as AsaSimulation, which provides the applications and necessary services to create and execute simulations. The second part comprises the interface applications, denominated AsaInterfaces, which provide tools for creating scenarios by listing all available components to be included and a library for interacting with the AsaSimulation module. Lastly, the third part is the analysis module, called AsaDataScience, which allows for post-processing and analysis of scenario executions.

Figure 1 displays a summary of the ASA architecture, and the following subsections provide details on all three primary ASA modules. All ASA applications use network communications, allowing processing to be distributed across multiple servers on a network.

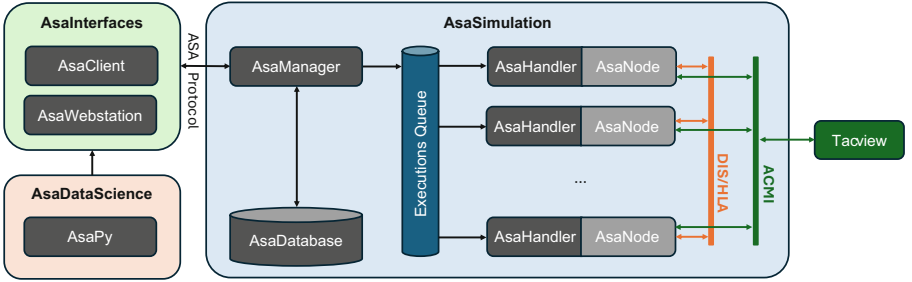


Fig. 1. ASA: modules and software applications.

### 2.1 AsaSimulation

The AsaSimulation module provides the necessary components for developing and executing a scenario simulation. It consists of applications, services, and libraries that help create agent models by developers and in the elaboration and simulation of scenarios by analysts.

The main features of this module are management and execution of simulations, dynamic loading of extension models, distribution of simulation executions, and management of simulation batch processing.

A permanent storage service called AsaDatabase keeps agent metadata, simulation scenarios, execution data, and analysis results. This service meets the storage demands of the AsaSimulation and AsaDataScience modules.

New agent models can be added to ASA by extending the interfaces and classes available in a library called AsaExtension. A new agent model that will be loaded into the AsaSimulation framework must provide its compiled source code (shared object file) and a JavaScript Object Notation (JSON) file that describes the parameters and components accepted by the model. AsaExtension also provides functionalities that allow extensions to store agent data in the AsaDatabase. This is done through the C++ macro `RECORD_ASA_CUSTOM_DATA("tag", agent)`, which receives as its first parameter a tag that identifies the agent's type and the agent object with the attributes to be stored in the AsaDatabase. Storing the state of the agents at each simulation step is optional, but it is essential if there is an interest in performing post-processing of the simulation data.

The AsaSimulation framework enables analysts to specify agent parameters (placeholders) during the simulation execution request, facilitating the execution of batch simulations. Analysts can request a batch execution from these simulation scenario templates by providing a list of initial conditions for the previously selected agent parameters. Each set of initial conditions, combined with the scenario template, generates an execution request that is allocated to run on a distributed processing node.

The distribution of simulation processing is done by dividing the responsibilities of simulating to three applications on a network: AsaManager, AsaHandler, and AsaNode. The following subsections detail each of these applications.

**AsaManager** is responsible for coordinating the processes to use ASA in a distributed manner. One of the essential functions of AsaManager is receiving and preprocessing execution requests, and automatically and transparently dispatching the requested simulation to be executed on an available node. The distribution of executions is performed using a queue service based on the Advanced Message Queuing Protocol (AMQP). AsaManager places a validated simulation execution request in the queue (Executions Queue) and, when available, an AsaHandler starts serving the request.

The use of queues to decouple direct management between request handling and execution enables the dynamic addition of new processing nodes, such as AsaHandler/AsaNode, to the system in response to increased demand for parallel execution. Because communication between AsaManager, ExecutionsQueue, and AsaHandler occurs over a network, applications can be distributed across different machines within a networked environment.

AsaManager incorporates a Representational State Transfer API (REST API) to enable interaction between other modules and the simulation functionalities, as well as access to stored data. This API implements the ASA protocol, which exposes AsaSimulation as a simulation service to other applications. It enables the submission, validation, execution, pausing, resumption, cancellation,

and monitoring of single- or batch-simulation tasks. Additionally, this protocol defines methods for accessing data from completed simulations and managing user data.

Access is restricted to authenticated and authorized users who are permitted to use the system's functionalities. For each simulation execution request, AsaManager verifies the validity of the scenario to be simulated and checks whether the authenticated user is authorized to use the listed components and has sufficient credit available. Within the ASA framework, these credits are referred to as AsaCoins, which serve as a means to account for user resource consumption. During each simulation execution, both the CPU time consumed and the disk space used to store the simulated agent data were measured. These two variables are then converted into an equivalent value in AsaCoins, which is subsequently debited from the user's account. The reference rate for a single AsaCoin unit was established using the execution of a standard benchmark scenario. Based on this reference, the CPU time conversion rate is 0.0137650 AsaCoin per second, and the storage conversion rate is 0.0141355 AsaCoin per kilobyte.

**AsaHandler** is responsible for monitoring the execution queue, converting requests, and supervising simulation execution. When an AsaHandler is available and a new execution request is placed in the queue by AsaManager, the AsaHandler retrieves the request, converts it into a format compatible with the AsaNode, initiates a new AsaNode process, and monitors its execution. Communication between AsaHandler and AsaNode is carried out via interprocess communication, enabling AsaHandler to oversee the simulation run. If the AsaNode terminates unexpectedly or a failure is detected by the AsaHandler, it will terminate the execution and will keep the AsaManager informed of the simulation status.

**AsaNode** is the simulation engine of the ASA platform, and its primary function is to process the simulation itself. It is an executable file obtained from the compilation of codes developed in MIXR and features developed by the ASA team, such as dynamically loading extensions and controlling the simulation's execution (pause, resume, stop, execution speed, etc.). It estimates how the scenario will evolve, considering the models incorporated in each agent present in the simulation. AsaNode can run on the same machine as the AsaManager application or in a clustered computing environment. This capability is essential when the user wants to simulate a set of scenarios, called batch, and the main difference between them is the initial configuration of each agent.

At predefined time intervals, AsaNode reports the progress of the simulation to AsaHandler, allowing this application, or AsaManager, to manage the execution process. If enabled by the execution requester, AsaNode can transmit data over the network using either the Distributed Interactive Simulation (DIS) protocol or the High-Level Architecture (HLA) standard, both of which are supported by the MIXR framework. DIS is a protocol designed for real-time exchange of

information between simulation entities in a distributed environment, while HLA is a general architecture that enables interoperability among different simulation systems through a shared runtime infrastructure [21]. To allow simulation data to be streamed for visualization in the Tacview flight analysis tool<sup>1</sup>, the AsaExtensions library implements the ACMI file format and its associated Real-time Telemetry Protocol [28].

## 2.2 AsaInterfaces

The AsaInterface module includes two tools: WebStation and AsaClient. WebStation provides a visual interface for building simulation scenarios, while AsaClient manages requests to the simulation manager and handles user authentication.

**WebStation** provides a graphical user interface (GUI) accessible to registered users through a web page developed with VueJS and Django. Analysts use the platform to construct simulation scenarios that include military symbols, geometric drawings, aeronautical charts, and digital terrain models. The interface displays how each scenario progresses during the simulation, offering an interactive environment for analysis and planning. Users can define performance metrics and examine the results after the simulation is completed. The platform also allows users to specify which messages should be stored in the database during execution. Furthermore, WebStation includes a validation feature that checks whether scenario components meet the model's requirements and ensures that all attributes and subcomponents are properly defined.

To support batch execution, WebStation allows users to configure input parameters that vary across multiple runs. After completing the scenario setup, users can download a JSON file containing the scenario data or save it to the database for future access. The interface also supports uploading JSON files, enabling users to edit and reuse existing scenarios.

An overview of the WebStation interface, including scenario construction and visualization features, is shown in Fig. 2.

**AsaClient** is a Python library designed to operate with the ASA protocol. It performs two primary functions: managing user authentication and handling communication with the simulation manager. AsaClient serves as an essential interface between the AsaDataScience environment and the Manager, enabling secure and structured access to simulation resources.

The library provides a set of high-level methods that allow analysts to interact with the core components of the simulation infrastructure. Through AsaClient, users can authenticate in the system, submit execution requests, and monitor simulation progress. AsaClient also supports database access for retrieving results, storing output, and managing simulation metadata.

<sup>1</sup> <https://www.tacview.net>.

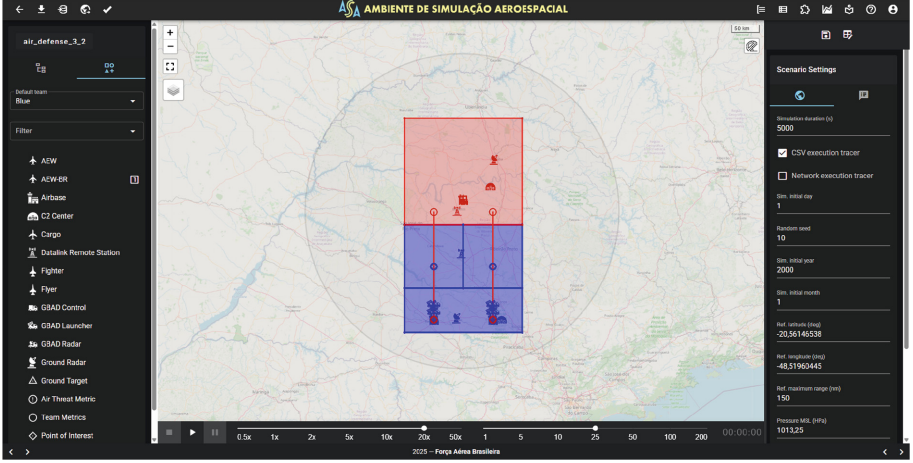


Fig. 2. WebStation: interface to create and visualize simulations.

### 2.3 AsaDataScience

The AsaDataScience module provides tools for understanding the factors that led to the simulation results, supporting analysts in identifying how performance, cost, and operational constraints evolve across military scenarios. It is composed of two main components: (i) the AsaPy Python library, designed to assist in the post-processing of simulation data; and (ii) a cloud-based computational environment with access to high-performance hardware and a JupyterHub interface for large-scale batch execution and interactive data analysis.

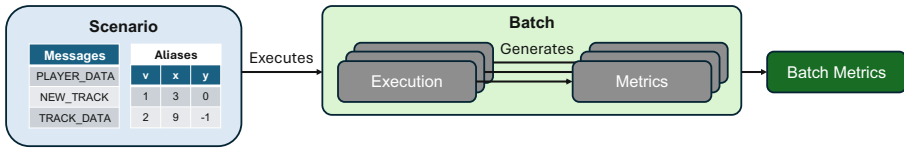
At the core of this module is AsaPy, a specialized Python library that provides a structured pipeline for simulation data analysis [15]. It integrates well-established methods for experiment planning, batch execution control, statistical analysis, and machine learning, enabling users to derive actionable insights with minimal programming effort. Its internal structure follows a typical analyst workflow and is composed of four main functional blocks, described in Table 1.

Running a simulation invariably results in the collection of massive data at every step. Once these data are adequately stored and structured in the Asa-Database, analysts can employ techniques such as data visualization, statistical testing, and model fitting to uncover insights from the simulations. AsaPy streamlines this process by automating routine steps and standardizing the analytical workflow.

A particularly important feature of AsaPy is its support for batch execution. In many military simulations, it is necessary to run hundreds or thousands of scenarios with varying input parameters [17]. AsaPy allows these executions to be divided into chunks, monitors the results of each chunk, and applies convergence-based early stopping criteria to avoid unnecessary runs. This improves both analytical efficiency and resource usage. Figure 3 illustrates this batch execution workflow.

**Table 1.** AsaPy functionalities and corresponding descriptions.

Functionality	Description
Design of Experiments	Generates input configurations to explore the simulation parameter space efficiently.
Execution Control	Manages simulation batches with chunking and early stopping based on convergence of key metrics.
Analysis	Offers tools for statistical analysis, data exploration, and visualization.
Prediction	Applies supervised and unsupervised machine learning models for regression and classification tasks.



**Fig. 3.** Batch execution flow supported by AsaPy. The process begins with the selection of aliases, which define the input parameters to be varied across simulations, and messages, which specify the simulation outputs to be collected. The scenario batch is then divided into sequential chunks. Each chunk is executed, and the selected outputs are used to compute batch-level metrics, which can support monitoring and early stopping of the execution process.

To support large-scale simulations, AsaDataScience includes a cloud-based computational environment equipped with high-performance hardware and a JupyterHub interface<sup>2</sup>. This setup enables analysts to manage simulation studies interactively through notebooks, providing both scripting flexibility and real-time visualization. JupyterHub also facilitates collaborative work by allowing multiple users to run batch executions, monitor progress, and analyze results in a shared and scalable environment, all without requiring direct access to low-level computing resources.

Overall, AsaDataScience serves as a unified environment for analyzing simulation data in defense applications, regardless of the simulation engine employed. In addition to supporting ASA-native data formats, it is also compatible with outputs from other frameworks, as demonstrated in [8].

### 3 Applications in Decision Support and Autonomy

ASA has been a flexible and useful platform for research in air combat operations, especially in areas related to decision support and autonomous systems. The studies listed below, in chronological order, demonstrate how ASA has been gradually adopted in increasingly complex and realistic scenarios.

<sup>2</sup> <https://jupyter.org/hub>.



The earliest study, [4], presented an artificial neural network model to support pilot situational awareness in Beyond Visual Range (BVR) air combat. Using simulation data, the system generated offensive and defensive assessments to aid decision-making during flight. This work was developed using the AEROGRAF Platform [27], which preceded ASA and influenced its initial design.

ASA was later used in [22] to explore the feasibility of agent-based modeling and simulation for assessing air defense capabilities in BVR engagements. This marked the first application of ASA in a strategic planning context.

In [7], over 50,000 missile launches were simulated to train a deep neural network for estimating the maximum launch range of a missile’s Weapon Engagement Zone (WEZ), achieving high predictive accuracy while reducing the need for further simulations. This study used the same missile model implemented in ASA.

In [6], 3,729 BVR engagements were simulated to train a supervised learning model aimed at supporting the decision of when to engage enemy aircraft during Defensive Counter Air (DCA) missions. The model was trained using simulation-derived operational metrics to capture relevant combat dynamics.

In [11], ASA/AEROGRAF was used to simulate 10,000 BVR combat scenarios, enabling the development of classifiers that improved pilot situational awareness. The models achieved high accuracy in distinguishing between offensive and defensive conditions.

The work in [3] focused on aircraft formation control using artificial potential fields and optimization techniques. ASA simulations helped identify configurations that improved formation coherence and mission success.

The study [24] employed ASA to optimize tactical unmanned aerial vehicle (UAV) formations under uncertainty in war-game scenarios. Metaheuristics were applied to maximize the probability of success against opposing forces.

In [12], ASA was used to propose an architecture for training a deep reinforcement learning agent capable of autonomously learning BVR tactics.

The work in [10] proposed an approach for analyzing surface-to-air missile engagement zones using prediction models to enable real-time estimation while reducing computational costs. The analysis was based on the same surface-to-air missile model implemented in ASA.

The work in [13] proposed new social navigation metrics to assess collaboration between human pilots and autonomous wingmen in air combat scenarios. It also outlined a validation experiment using ASA to simulate mixed human-autonomous formations.

In [30], a defensive “winding maneuver” was proposed to improve aircraft survivability against surface-to-air missiles. Developed using ASA, the maneuver was optimized with a genetic algorithm and validated through simulations, statistical analysis, and operational metrics.

The work in [23] proposed machine learning models to approximate ASA simulation outcomes in stochastic contexts, significantly reducing overall simulation time.

The study in [16] introduced a probabilistic kill estimation model for air-to-air missiles based on ASA simulations and stochastic target modeling. It employed the same missile structure used in ASA.

Finally, [5, 14] used pilot data from ASA/AEROGRAF to train imitation learning models for replicating pop-up attack maneuvers. A variational autoencoder was employed to generate synthetic samples and augment the dataset, improving the robustness and performance of the models.

These studies collectively demonstrate the flexibility and effectiveness of ASA as a simulation tool for operational analysis, autonomy development, and tactical decision support in air combat environments.

## 4 Conclusion and Future Work

In this work, we presented a high-level overview of the ASA simulation framework, developed by IEAv since 2018, with the primary objective of supporting the evaluation of military operational scenarios relevant to FAB. The platform offers several distinguishing features, including the management of multiple simulation machines across one or more computers, as well as the dynamic loading .so files at runtime, batch execution of simulations, and the AsaDataScience module, an integrated data analysis platform tailored for military operational studies. Furthermore, we highlighted recent works that have employed ASA as a simulation tool to support air combat applications.

For future work, we plan to release part of the ASA source code, including its general architecture, to a selected group of organizations. This controlled release aims to encourage the development of diverse applications within the same simulation platform while ensuring proper control and security. Additionally, ASA is expected to evolve into a Simulation-as-a-Service (SimaaS) tool, supporting a wide range of simulation demands in the defense and aerospace domains [9]. This initiative aims to promote greater interoperability among government, academia, and industry.

From a technical perspective, several enhancements are under consideration. First, we aim to enhance scalability through parallel processing by enabling the flexible addition of simulation nodes as computational demands increase. Another area of focus is observability: ASA will incorporate mechanisms to monitor resource usage (such as CPU, memory, and disk), track feature utilization (identifying which components and libraries are used most frequently), and detect failures. These capabilities will enable better instrumentation, increased failure tolerance, and enhanced platform operational visibility.

We also aim to introduce accountability mechanisms to estimate and control the computational cost of each simulation, allowing resource usage to be better quantified and managed. Regarding the core simulation engine, currently based on an object-oriented paradigm with deep inheritance hierarchies, we are exploring a transition to a more modular and maintainable design. One promising alternative is the adoption of the entity-component-system (ECS) paradigm, which can improve model readability and simplify the integration of new components [26].

Additionally, improvements are planned for the internal structure of the AsaPy library, with efforts to reorganize its modules for better maintainability and extensibility. Finally, we expect to adapt the framework to support more modular and flexible decision-making processes in autonomous agent models by developing an artificial intelligence server in Python that integrates a rich ecosystem of specialized tools and libraries.

**Acknowledgments.** This work has been supported by the Funding Authority for Studies and Projects (FINEP)—Finance Code 01.20.0196.00/2824/20.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References

1. Clive, P.D., Johnson, J.A., Moss, M.J., Zeh, J.M., Birkmire, B.M., Hodson, D.D.: Advanced framework for simulation, integration and modeling (AFSIM). In: The 13th International Conference on Scientific Computing (CSC'15). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), Las Vegas, NV, USA, p. 73 (2015)
2. Costa, A.N., Dantas, J.P.A., Scukins, E., Medeiros, F.L.L., Ögren, P.: Simulation and machine learning in beyond visual range air combat: a survey. *IEEE Access* **13**, 76755–76774 (2025). <https://doi.org/10.1109/ACCESS.2025.3563811>
3. Costa, A.N., Medeiros, F.L., Dantas, J.P.A., Geraldo, D., Soma, N.Y.: Formation control method based on artificial potential fields for aircraft flight simulation. *SIMULATION* **98**(7), 575–595 (2022). <https://doi.org/10.1177/00375497211063380>, <https://doi.org/10.1177/00375497211063380>
4. Dantas, J.P.A.: Apoio à Decisão para o Combate Aéreo Além do Alcance Visual: Uma Abordagem por Redes Neurais Artificiais. Master's, Instituto Tecnológico de Aeronáutica, São José dos Campos, SP, Brazil (2018)
5. Dantas, J.P.A.: Autonomous pop-up attack maneuver using imitation learning. In: Proceedings of the Winter Simulation Conference, PhD Colloquium, Orlando, FL, USA. IEEE (2024)
6. Dantas, J.P.A., Costa, A.N., Geraldo, D., Maximo, M.R.O.A., Yoneyama, T.: Engagement decision support for beyond visual range air combat. In: Proceedings of the 2021 Latin American Robotics Symposium, 2021 Brazilian Symposium on Robotics, and 2021 Workshop on Robotics in Education, pp. 96–101. IEEE, Natal, RN, Brazil (2021)
7. Dantas, J.P.A., Costa, A.N., Geraldo, D., Maximo, M.R.O.A., Yoneyama, T.: Weapon engagement zone maximum launch range estimation using a deep neural network. In: Britto, A., Valdivia Delgado, K. (eds.) BRACIS 2021. LNCS (LNAI), vol. 13074, pp. 193–207. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-91699-2\\_14](https://doi.org/10.1007/978-3-030-91699-2_14)
8. Dantas, J.P.A., Costa, A.N., Medeiros, F.L.L., Geraldo, D., Maximo, M.R.O.A., Yoneyama, T.: Supervised machine learning for effective missile launch based on beyond visual range air combat simulations. In: Proceedings of the Winter Simulation Conference. WSC '22, Singapore. IEEE (2022)

9. Dantas, J.P.A., Geraldo, D., Costa, A.N., Maximo, M.R.O.A., Yoneyama, T.: ASA-SimaaS: Advancing digital transformation through simulation services in the brazilian air force. In: Simpósio de Aplicações Operacionais em Áreas de Defesa (SIGE2023), p. 6. Instituto Tecnológico de Aeronáutica (ITA), São José dos Campos, Brazil (2023). [https://www.sige.ita.br/edicoes-anteriores/2023/st/235455\\_1.pdf](https://www.sige.ita.br/edicoes-anteriores/2023/st/235455_1.pdf)
10. Dantas, J.P.A., Geraldo, D., Medeiros, F.L.L., Maximo, M.R.O.A., Yoneyama, T.: Real-time surface-to-air missile engagement zone prediction using simulation and machine learning. In: Proceedings of the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC), Orlando, FL, USA. National Training and Simulation Association (NTSA) (2023)
11. Dantas, J.P.A., Maximo, M.R.O.A., Costa, A.N., Geraldo, D., Yoneyama, T.: Machine learning to improve situational awareness in beyond visual range air combat. IEEE Latin Am. Trans. **20**(8) (2022). <https://latam.t.ieee.org/index.php/transactions/article/view/6530>
12. Dantas, J.P.A., Maximo, M.R.O.A., Yoneyama, T.: Autonomous agent for beyond visual range air combat: a deep reinforcement learning approach. In: Proceedings of the 2023 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS'23), pp. 13–24. SIGSIM-PADS'23, New York, NY, USA. Association for Computing Machinery (2023). <https://doi.org/10.1145/3573900.3593631>
13. Dantas, J.P.A., Maximo, M.R.O.A., Yoneyama, T.: Loyal Wingman Assessment: Social Navigation for Human-Autonomous Collaboration in Simulated Air Combat. In: Proceedings of the 38th ACM SIGSIM Conference on Principles of Advanced Discrete Simulation. SIGSIM-PADS '24, New York, NY, USA, pp. 61–62. Association for Computing Machinery (2024). <https://doi.org/10.1145/3615979.3662149>, <https://doi.org/10.1145/3615979.3662149>
14. Dantas, J.P.A., Maximo, M.R.O.A., Yoneyama, T.: Autonomous aircraft tactical pop-up attack using imitation and generative learning. IEEE Access **13**, 81204–81217 (2025). <https://doi.org/10.1109/ACCESS.2025.3567186>
15. Dantas, J.P.A., et al.: AsaPy: a python library for aerospace simulation analysis. In: Proceedings of the 38th ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, pp. 15–24. SIGSIM-PADS '24, New York, NY, USA. Association for Computing Machinery (2024). <https://doi.org/10.1145/3615979.3656063>
16. Dantas, J.P., Costa, A.N., Geraldo, D., Maximo, M.R., Yoneyama, T.: PoKER: a probability of kill estimation rate model for air-to-air missiles using machine learning on stochastic targets. J. Def. Model. Simul. **0**(0), 15485129241309675 (2025). <https://doi.org/10.1177/15485129241309675>
17. Gill, J., Grieger, D., Wong, J., Chau, M.: Regression analysis, multiple comparisons and ranking sensitivity. In: Proceedings of the 2018 Winter Simulation Conference (WSC), Gothenburg, Sweden, pp. 3789–3800. IEEE (2018). <https://doi.org/10.1109/WSC.2018.8632534>, <https://www.informs-sim.org/wsc18papers/includes/files/338.pdf>
18. Han, Y., et al.: Deep relationship graph reinforcement learning for multi-aircraft air combat. In: 2022 International Joint Conference on Neural Networks (IJCNN), Padua, Italy, pp. 1–8. IEEE (2022). <https://doi.org/10.1109/IJCNN55064.2022.9892208>, <https://ieeexplore.ieee.org/abstract/document/9892208>, ISSN: 2161-4407
19. Hill, R.R., Tolk, A., Hodson, D.D., Millar, J.R.: Open challenges in building combat simulation systems to support test, analysis and training. In: 2018 Winter

- Simulation Conference (WSC), Gothenburg, Sweden, pp. 3730–3741. Association for Computing Machinery (2018). <https://doi.org/10.1109/WSC.2018.8632233>
20. Hodson, D.D., Gehl, D.P.: The mixed reality simulation platform (MIXR). In: The 16th International Conference on Scientific Computing (CSC’18). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), Las Vegas, USA, p. 73 (2018)
21. Kuhl, F., Weatherly, R., Dahmann, J.: Creating Computer Simulation Systems: An Introduction to the High Level Architecture. Prentice Hall, Upper Saddle River (1999)
22. Kuroswiski, A.R.: Modelagem e Simulação Baseada em Agentes como Ferramenta de Apoio à Avaliação de Capacidades de Defesa Aérea. Master’s, Instituto Tecnológico de Aeronáutica, São José dos Campos, SP, Brazil (2020)
23. Lima, L., Giannico, R., Brito, D.D., Dantas, A., Dantas, J.: Aprendizado de Máquina para a Otimização da Obtenção de Resultados em Simulações de Defesa Aeroespacial. In: Congresso Acadêmico sobre Defesa Nacional, Rio de Janeiro, RJ. Escola Superior de Defesa (ESD) (2024). <https://doi.org/10.5281/zenodo.13334319>
24. Lima Filho, G.M., Kuroswiski, A.R., Medeiros, F.L.L., Voskuijl, M., Monsuur, H., Passaro, A.: Optimization of unmanned air vehicle tactical formation in war games. *IEEE Access* **10**, 21727–21741 (2022). <https://doi.org/10.1109/ACCESS.2022.3152768>
25. MAK Technologies: Vr-forces. <https://www.mak.com/mak-one/apps/vr-forces/> (2023). Accessed 15Apr 2023
26. Muratet, M., Garbarini, D.: Accessibility and serious games: what about entity-component-system software architecture? In: Marfisi-Schottman, I., Bellotti, F., Hamon, L., Klemke, R. (eds.) *GALA 2020. LNCS*, vol. 12517, pp. 3–12. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-63464-3\\_1](https://doi.org/10.1007/978-3-030-63464-3_1)
27. Petersen, J.F., Aquino, M.R.C., Salles, R.N.: Plataforma AEROGRAF: um SIG voltado para a Força Aérea. *Revista Spectrum*, n. 11, Comando-Geral de Operações Aéreas (2008)
28. Raia Software Inc.: Documentation repository of tacview: Advanced flight analysis tool (2025). <https://www.tacview.net/documentation/index/en/>. Accessed 26 May 2025
29. Ternion: FLAMES Development Suite (2023). <https://flamesframework.com/>. Accessed 26 May 2025
30. Viscardi Filho, M., Dantas, J., Geraldo, D., Pássaro, A.: Manobra winding: defesa contra mísseis passivos e semiativos superfície-ar. *Spectrum - The Journal of Operational Applications in Defense Areas* **25**(1), 12–17 (out 2024). <https://doi.org/10.55972/spectrum.v25i1.403>, <https://spectrum.ita.br/index.php/spectrum/article/view/403>